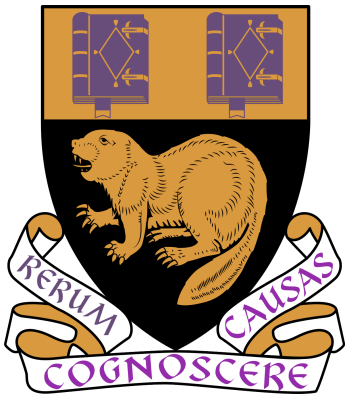
Three Essays in Bayesian Nonparametric Machine Learning



Yirui Liu

The Department of Statistics

London School of Economics and Political Science

A thesis submitted for the degree of

Doctor of Philosophy

July 2022

Declaration

I certify that the thesis I have presented for examination for the PhD degree of the London School of Economics and Political Science is solely my own work other than where I have clearly indicated that it is the work of others (in which case the extent of any work carried out jointly by me and any other person is clearly identified in it).

I confirm that Chapter 2 was jointly co-authored with Dr Xinghao Qiao and Dr Jessica Lam. Chapter 3 was jointly co-authored with Dr Xinghao Qiao, Dr Liying Wang and Dr Jessica Lam. Chapter 4 was jointly co-authored with Dr Xinghao Qiao, Dr Yulong Pei and Dr Liying Wang.

Acknowledgements

I want to express my sincere appreciation to Dr Xinghao Qiao for his consistent support, patient guidance, and professional advice. I am also thankful to Professor Qiwei Yao, Professor Milan Vojnovic, Professor Angelos Dassios, Professor Umut Cetin, and Ms Penny Montague for their encouragement and support. I am grateful to my family for their love.

Abstract

There is a wide application of Bayesian nonparametric machine learning to a variety of fields, such as bioinformation, language processing, computer vision, network analysis, economics and finance. In Bayesian nonparametric models, Dirichlet process, Indian buffet process, Gaussian process, and other priors are used to obtain an infinitely dimensional prior distribution and hence to infer the dimensionality of parameter space in a data-adaptive manner. In this thesis, I present a variational inference framework for hierarchical Bayesian nonparametric models and develop state-of-the-art models that combine Bayesian nonparametrics and deep learning for graph data and sequential data.

First, I develop a novel inference method for the hierarchical Bayesian nonparametric models, especially for the Dirichlet process model. Current variational inference methods for hierarchical Bayesian nonparametric models can neither characterize the correlation structure among latent variables due to the mean-field setting, nor infer the true posterior dimension because of the universal truncation. To overcome these limitations, I propose the conditional and adaptively truncated variational inference method (CATVI) by maximizing the nonparametric evidence lower bound and integrating Monte Carlo into the variational inference framework. CATVI enjoys several advantages over traditional methods, including a

smaller divergence between variational and true posteriors, reduced risk of underfitting or overfitting, and improved prediction accuracy. Empirical studies on three large datasets reveal that CATVI applied in Bayesian nonparametric topic models substantially outperforms competing models, providing lower perplexity and clearer topic-words clustering.

Moreover, I develop a methodology of using Bayesian nonparametric to improve the performance of deep graph neural networks. Training deep graph neural networks (GNNs) poses a challenging task, as the performance of GNNs may suffer from the number of hidden message-passing layers. The literature has focused on the proposals of over-smoothing and under-reaching to explain the performance deterioration of deep GNNs. I propose a new explanation for such deteriorated performance phenomenon, mis-simplification, that is, mistakenly simplifying graphs by preventing self-loops and forcing edges to be unweighted. I show that such simplifying can reduce the potential of message-passing layers to capture the structural information of graphs. In view of this, I propose a new framework, edge enhanced graph neural network (EEGNN). EEGNN uses the structural information extracted from the proposed Dirichlet mixture Poisson graph model, a Bayesian nonparametric model for graphs, to improve the performance of various deep message-passing GNNs. Experiments over different datasets show that our method achieves considerable performance increase compared to baselines.

Finally, I present Deep Functional Factor Model (DF^2M), a Bayesian nonparametric model for analyzing high-dimensional functional time series. The DF^2M makes use of the Indian Buffet Process and the multi-task

Gaussian Process with a deep kernel function to capture non-Markovian and nonlinear temporal dynamics. Unlike many black-box deep learning models, the DF^{2}M provides an explainable way to use neural networks by constructing a factor model and incorporating deep neural networks within the kernel function. Additionally, I develop a computationally efficient variational inference algorithm for inferring the DF^{2}M. Empirical results from four real-world datasets demonstrate that the DF^{2}M offers better explainability and superior predictive accuracy compared to conventional deep learning models for high-dimensional functional time series.

Contents

Chapter 1

Introduction

Bayesian nonparametric models provide a flexible framework for modeling complex patterns in datasets. Unlike parametric Bayesian methods, which assume a finite-dimensional probability distribution for the priors in a particular format, Bayesian nonparametric models place a prior distribution over an infinite-dimensional space of functions (Ghosal and Van der Vaart, 2017). This allows for the modeling of complex data without the need to specify a particular parametric form. Therefore, Bayesian nonparametric models are widely used in various machine learning fields, such as natural language processing, computer vision, graph and network modeling, federated learning and sequential decision making (Sudderth and Jordan, 2009; Choi and Kim, 2012; Williamson, 2016; Yurochkin et al., 2019). The commonly used infinite-dimensional priors in Bayesian nonparametric models include the Dirichlet process (Ferguson, 1973), Indian buffet process (Griffiths and Ghahramani, 2011) and Gaussian process (Williams and Rasmussen, 2006), etc.

In particular, recent advances in Markov Chain Monte Carlo techniques and variational inference methods have facilitated their application to increasingly larger and more complicated datasets (Kroese et al., 2011; Hoffman et al., 2013). For example, nonparametric Bayesian deep learning, which combines the strengths of Bayesian nonparametrics and deep learning, has opened up a new direction for research. In

this example, nonparametric Bayesian methods are used to impose priors over the parameters of deep learning models, enabling them to adapt their complexity to the data, enhance their interpretability, and provide more robust prediction (Ghahramani, 2015). Despite these advances, many challenges remain, such as developing efficient nonparmetric inference algorithms and improving explainability and robustness, making Bayesian nonparametric machine learning a vibrant and dynamic field of research.

This thesis delves into the heart of Bayesian nonparametric models including the Dirichlet process, Indian buffet process, and Gaussian process. It presents a novel variational inference framework and introduces cutting-edge models that combine Bayesian nonparametrics and deep learning for graph and sequential data. The thesis first unravels Conditional and Adaptively Truncated Variational Inference (CATVI), a new inference method for hierarchical Bayesian nonparametric models, especially for the hierarchical Dirichlet process model, overcoming the limitations of current variational inference methods. Further, it introduces a new framework, the Edge Enhanced Graph Neural Network (EEGNN), exploring the enhancement of deep graph neural networks performance using Bayesian nonparametrics. Finally, it discusses the Deep Functional Factor Model (DF^2M), a Bayesian nonparametric model for analyzing high-dimensional functional time series, offering higher predictive accuracy and better explainability compared to traditional deep learning models. This thesis sheds light on the potential of Bayesian nonparametric models combined with modern machine learning algorithms for the analysis of complex datasets.

Chapter 2

Variational Inference for Bayesian Nonparametric Models

This chapter is dedicated to an article published at the 25th International Conference on Artificial Intelligence and Statistics, available online at `https://proceedings.mlr.press/v151/liu22d.html`.

CATVI: Conditional and Adaptively Truncated Variational Inference for Hierarchical Bayesian Nonparametric Models

Yirui Liu
London School of Economics

Xinghao Qiao
London School of Economics

Jessica Lam
JP Morgan Chase & Co.

Abstract

Current variational inference methods for hierarchical Bayesian nonparametric models can neither characterize the correlation structure among latent variables due to the mean-field setting, nor infer the true posterior dimension because of the universal truncation. To overcome these limitations, we propose the conditional and adaptively truncated variational inference method (CATVI) by maximizing the nonparametric evidence lower bound and integrating Monte Carlo into the variational inference framework. CATVI enjoys several advantages over traditional methods, including a smaller divergence between variational and true posteriors, reduced risk of underfitting or overfitting, and improved prediction accuracy. Empirical studies on three large datasets reveal that CATVI applied in Bayesian nonparametric topic models substantially outperforms competing models, providing lower perplexity and clearer topic-words clustering.

1 INTRODUCTION

Hierarchical Bayesian nonparametric (HBNP) models are widely used in bioinfomatics, language processing, computer vision and network analysis (Sudderth and Jordan, 2009; Caron and Fox, 2017; Williamson, 2016; Yurochkin et al., 2019). A major benefit of HBNP models is their ability to relax the fixed dimension assumption in parametric models. For example, in natural language processing, hierarchical Dirichlet process (HDP) model (Teh et al., 2006) replaces the finite-dimensional Dirichlet distribution in latent Dirichlet allocation (LDA) with a countable-dimensional Dirichlet process (DP). This is done by regarding the number of topics as a random variable that can be inferred from the data, rather than as a parametric value (Blei et al., 2003).

However, it is much harder to implement HBNP models than their parametric counterparts. In particular, due to a HBNP model's infinite-dimensional nature, a finite-dimensional truncation is needed to approximate the posterior. Yet, the selection of the optimal truncation level poses several challenges. On one hand, the traditional Markov chain Monte Carlo (MCMC) methods (Teh et al., 2006) can produce an adaptive selection of the truncated dimension, but they are not computationally scalable especially for big data. On the other hand, standard variational inference methods (Teh et al., 2008; Wang et al., 2011; Hoffman et al., 2013; Roychowdhury and Kulis, 2015; Xu et al., 2019) can accelerate the computation, but they suffer from a universal selection of the truncation level by truncating the dimension of all latent variables to a prespecified value. Using a prespecified value is problematic, because a subjective selection of the fixed truncation level can make the model prone to overfitting or underfitting, leading to low predictive accuracy. These existing challenges in universal truncation contradict the motivation and advantages of using HBNP models.

In this paper, we propose a general framework, called conditional and adaptively truncated variational inference (CATVI), to infer HBNP models in the following steps. First, we convert the inference problem to an optimization task of maximizing our proposed nonparametric evidence lower bound based on finite partitions. Second, we introduce a conditional setting when factorizing variational distributions by conditioning variables in the middle layers on two adjacent layers. Third, to handle big data, we develop a stochastic variational inference framework (Blei et al., 2017) under our conditional setting. Finally, we obtain empirical distributions from Monte Carlo sampling of local latent variables, which are then used to update the variational parameters for the global latent vari-

Proceedings of the 25$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

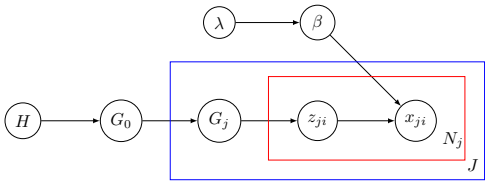


Figure 1: The HBNP models. The blue and red boxes correspond to J and N_j replicates, respectively.

ables. This enables us to truncate the dimension of the latent variational distributions to that of the empirical distribution.

Our proposed method benefits from both the inferential accuracy of Monte Carlo sampling and the computational efficiency of variational inference. First, our method rebuilds the correlation structure and hence attains a smaller Kullback–Leibler (KL) divergence between the variational distribution and the true posterior. Such procedure removes the unrealistic mean-field assumption, and searches for an optimal variational distribution over a wider family. Second, it adjusts the dimension of variation distributions, which converges to a stable level balancing the goodness-of-fit and model complexity. With these advantages, CATVI provides an adaptive selection of the truncated dimension, reducing the risk of overfitting or underfitting, while also enabling more accurate predictions without sacrificing the computational efficiency. Specific to the inference for the HDP model, CATVI enjoys several advantages over existing methods (Teh et al., 2006; Hoffman et al., 2013; Wang and Blei, 2012; Bryant and Sudderth, 2012), see Section 6 for our detailed discussion.

2 BACKGROUND: HBNP MODELS

As a subclass of Bayesian nonparametric models, HBNP models extend the simplicity of using random measures (see Appendix A) as priors to the following hierarchical structure,

$$\begin{aligned} G_0|H \sim P(H), \quad \beta|\lambda \sim p(\beta|\lambda), \quad G_j|G_0 \sim R(G_0), \\ z_{ji}|G_j \sim G_j, \quad x_{ji}|z_{ji} \sim f(x_{ji}|\beta, z_{ji}), \end{aligned} \tag{1}$$

for $j = 1, \ldots, J, i = 1, \ldots, N_j$, as illustrated in Figure 1. In the top layer, $G_1, \ldots, G_J$ are generated from a random measure R with common base measure G_0, while in the bottom layer, G_0 itself is a realization of random measure P with base measure H. To ensure exchangeability, $G_1, \ldots, G_J$ are assumed to be identical and independent given G_0. Each local latent variable z_{ji} is sampled from G_j independently. Finally, the global parameter β is assigned a prior $p(\beta|\lambda)$, and the observation x_{ji} is generated from a likelihood function f, parameterized by both global latent variable β and local latent variables z_{ji}.

In topic modelling, the HDP model (Teh et al., 2006) uses a DP for both P and R in (1) as,

$$G_0|H \sim \mathrm{DP}(\alpha H), \quad G_j|G_0 \sim \mathrm{DP}(\gamma G_0), \tag{2}$$

where α, γ are concentration parameters, and H, G_0 are normalized based measures (see Appendix A). Suppose a corpus has J documents, each document j has N_j words, and each word is chosen from a vocabulary with W terms. Specifically, $G_0 = \sum_{k=1}^{\infty} G_{0k}\delta_{\phi_k}$ is generated from the distribution $\mathrm{DP}(\alpha H)$, and for each document j, a topic proportion, defined as $G_j = \sum_{k=1}^{\infty} G_{jk}\delta_{\phi_k}$, is independently sampled from the distribution $\mathrm{DP}(\gamma G_0)$. For each topic k, the distribution of words over vocabulary is sampled from a W-dimensional Dirichlet distribution parameterized by η, $\beta_k = (\beta_{k,1}, \cdots, \beta_{k,W})^{\mathrm{T}} \sim \mathrm{Dir}(\eta)$. For each word i in document j, a topic assignment $z_{ji} = \phi_k$ is chosen from $z_{ji} \sim \mathrm{Multinomial}(G_j)$, where ϕ_k represents topic k. Finally, the observation x_{ji} is generated from the assigned topic and the corresponding within-topic word distribution, $x_{ji}|\{z_{ji} = \phi_k\} \sim \mathrm{Multinomial}(\beta_k)$.

The necessity to let G_0 be atomic can be shown in the HDP model. If $G_1, \ldots, G_J$ are sampled from a Dirichlet process with a diffuse base measure instead of an atomic G_0, $G_1, \ldots, G_J$ will not share any support almost surely, and thus none of the topics being shared across the documents. However, for a general HBNP model, as long as G_0 is atomic, it is not necessary to restrict the prior for G_0 to be a Dirichlet process or a probability random measure. For example, the ΓDP model, which has the following structure,

$$G_0|H \sim \Gamma\mathrm{P}(\alpha H), \quad G_j|G_0 \sim \mathrm{DP}(G_0), \tag{3}$$

allows for more flexibility by removing the constraint on the concentration parameter in the top layer. Other choices of the prior for G_0 include beta process, stable process and inverse Gaussian process (Ghosal and Van der Vaart, 2017).

To infer the HBNP models, we set up the theoretical foundations for nonparametric KL divergence and evidence lower bound, and then propose a novel methodology in the following Sections 3 and 4.

3 NONPARAMETRIC EVIDENCE LOWER BOUND

3.1 KL Divergence between Random Measures

The object of variational inference is to minimize the KL divergence between the variational distribution

and the true posterior. For two infinite-dimensional random measures, their KL divergence is well defined even though an infinite-dimensional density function does not exist in a conventional sense. Given two random measures P and Q from $(\Theta, \mathcal{M})$ into $(\Omega, \mathcal{F})$, the Radon–Nikodym derivative dQ/dP exits if Q is absolutely continuous with respect to P. Their KL divergence is defined as

$$\mathcal{KL}(Q \parallel P) = \int_{\Theta} \log(dQ/dP) dQ,$$

which is intractable due to the infinite-dimensional integral (Matthews et al., 2016). We have developed a new approach to calculate it using the limit superior of the divergence between corresponding finite-dimensional induced measures, that is,

$$\mathcal{KL}(Q \parallel P) = \limsup_{\varOmega} \mathcal{KL}(q^{\varOmega} \parallel p^{\varOmega}), \tag{4}$$

where $p^{\varOmega}$ and $q^{\varOmega}$ are respectively induced measures from P and Q on a finite partition $\varOmega = (A_1, \ldots, A_n)$, such that $p^{\varOmega}(A_i) = P(A_i)$ and $q^{\varOmega}(A_i) = Q(A_i)$ for each $A_i \in \varOmega$. With an induced random variable $Z^{\varOmega}\colon \Theta \to \mathbb{R}^n$, we can also denote the induced measures by $p(Z^{\varOmega})$ and $q(Z^{\varOmega})$. The result in (4) is justified in Appendix C.1. We use the following two examples to illustrate (4).

Example 1 For Poisson processes $P = \mathrm{PP}(\Lambda + b\delta_{\phi})$ and $Q = \mathrm{PP}(\Lambda + a\delta_{\phi})$, where Λ is the intensity function defined on Ω, $a, b \in \mathbb{R}^+$, and δ_{ϕ} is a Dirac function at point $\phi \in \Omega$. Under partition $\varOmega = (\phi, \Omega/\phi)$, the limit superior in (4) is achieved, that is,

$$\mathcal{KL}(Q \parallel P) = \mathcal{KL}\big(\mathrm{Pois}(a) \parallel \mathrm{Pois}(b)\big),$$

where $\mathrm{Pois}(a)$ is the Poisson distribution with intensity a.

Example 2 For Dirichlet processes $P = \mathrm{DP}(\alpha H + \sum_{i=1}^{n} b_i \delta_{\phi_i})$ and $Q = \mathrm{DP}(\alpha H + \sum_{i=1}^{n} a_i \delta_{\phi_i})$, where H is the base measure, α is the concentration parameter, $a_i, b_i \in \mathbb{R}^+$ and $\phi_i \in \Omega$ for $i = 1, \ldots n$. Similarly, under the partition $\varOmega = \big(\phi_1, \ldots, \phi_n, \Omega/\{\phi_i\}_{i=1}^n\big)$,

$$\mathcal{KL}(Q \parallel P) =$$
$$\mathcal{KL}\big(\mathrm{Dir}(\alpha, a_1, \ldots, a_n) \parallel \mathrm{Dir}(\alpha, b_1, \ldots, b_n)\big),$$

where $\mathrm{Dir}(\alpha, a_1, \ldots, a_n)$ is the Dirichlet distribution with parameters $\alpha, a_1, \ldots, a_n$.

With the KL divergence between random measures represented under a finite partition, we can then define the nonparametric counterpart of evidence lower bound below.

3.2 Nonparametric Evidence Lower Bound

The parametric variational inference algorithm uses a finite-dimensional variational distribution to approximate the posterior by maximizing the evidence lower bound (Blei et al., 2017). In contrast, HBNP models uses a random measure for the variational distribution, due to the infinite dimensionality of latent variables. We propose a general inference framework for HBNP models by maximizing the nonparametric evidence lower bound (NPELBO), defined as the limit inferior of the parametric evidence lower bound under a finite partition, $\liminf_{\varOmega}(\mathrm{ELBO}^{\varOmega})$, that is

$$\liminf_{\varOmega} \big\{ \mathrm{E}_{q(Z^{\varOmega})} \log p(X, Z^{\varOmega}) - \mathrm{E}_{q(Z^{\varOmega})} \log q(Z^{\varOmega}) \big\}, \tag{5}$$

where $p(X, Z^{\varOmega})$ and $q(Z^{\varOmega})$ correspond to the induced measures from the joint distribution and the variational distribution on $\varOmega$, and where Z and X are the observations and latent variables, respectively. Moreover, given the KL divergence between random measures in (4), in Appendix C.2 we show that

$$\mathcal{KL}\big(Q(Z) \parallel P(Z|X)\big) + \mathrm{NPELBO} = \log p(X). \tag{6}$$

This demonstrates the equivalence between maximizing the NPELBO in (5) and minimizing the KL divergence between the variational distribution $Q(Z)$ and the true posterior $P(Z|X)$. The task of maximizing the NPELBO is general and can be applied broadly within Bayesian nonparametrics. To simplify notation, we will use $p(\cdot)$ and $q(\cdot)$ to denote the true and variational distributions, respectively, where the context is clear. To infer HBNP models, we aim to maximize the defined NPELBO, while truncating the dimension of variational distribution adaptively as follows.

4 METHODOLOGY

CATVI adopts the stochastic variational inference framework (Hoffman et al., 2013), where the computation is accelerated by selecting a small batch of data and updating variational parameters with an unbiased random gradient. We first build the foundation of conditional variational inference as follows.

4.1 Conditional Variational Inference

Conditional setting HBNP models in (1) contain global latent variable β, local latent variables $\boldsymbol{z}$, global prior G_0, local priors $\boldsymbol{G}_{[J]}$ and observations $\boldsymbol{x}$, where $\boldsymbol{z} = \{\boldsymbol{z}_j\}_{j=1}^{J}$, $\boldsymbol{z}_j = \{z_{ji}\}_{i=1}^{N_j}$, $\boldsymbol{x} = \{\boldsymbol{x}_j\}_{j=1}^{J}$, $\boldsymbol{x}_j = \{x_{ji}\}_{i=1}^{N_j}$ and $\boldsymbol{G}_{[J]} = \{G_j\}_{j=1}^{J}$. We aim to find the variational distribution to maximize the NPELBO. In contrast to traditional approaches under the mean-field setting, we factorize the variational distribution

as

$$q(\beta, \boldsymbol{z}, G_0, \boldsymbol{G}_{[J]}) = q(\beta)q(G_0)\prod_{j=1}^{J} q(G_j|G_0, \boldsymbol{z}_j)\prod_{i=1}^{N_j} q(z_{ji}), \tag{7}$$

in the sense of the probability law. Such conditional design facilitates the recovery of the dependence structure among G_0, $\boldsymbol{G}_{[J]}$ and $\boldsymbol{z}$.

Combing (5) and (7), we seek to maximize the following NPELBO:

$$\begin{aligned}\liminf_{\varrho} \Big\{ & \mathrm{E}_{q(\beta,\boldsymbol{z},G_0^\varrho,\boldsymbol{G}_{[J]}^\varrho)} \log p(\boldsymbol{x}, \beta, \boldsymbol{z}, G_0^\varrho, \boldsymbol{G}_{[J]}^\varrho) \\ & - \sum_{j=1}^{J} \mathrm{E}_{q(G_0^\varrho)}\mathrm{E}_{q(\boldsymbol{z}_j)}\mathrm{E}_{q(G_j^\varrho|G_0^\varrho,\boldsymbol{z}_j)} \log q(G_j^\varrho|G_0^\varrho, \boldsymbol{z}_j) \\ & - \mathcal{H}\big(q(G_0^\varrho)\big) - \mathcal{H}\big(q(\beta)\big) - \sum_{j=1}^{J}\sum_{i=1}^{N_j} \mathcal{H}\big(q(z_{ji})\big) \Big\},\end{aligned} \tag{8}$$

where the entropy $\mathcal{H}\big(q(\cdot)\big) = E_{q(\cdot)} \log q(\cdot)$ and Ω is a partition of the sample space Ω for G_0 and $\boldsymbol{G}_{[J]}$.

Conditional variational distribution To maximize the NPELBO in (8), we first compute the optimal variational distribution of G_j given G_0 and $\boldsymbol{z}_j$ for each j. As $p(\boldsymbol{x}, \beta, \boldsymbol{z}, G_0^\varrho, \boldsymbol{G}_{[J]}^\varrho) = p(G_0^\varrho, \boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z})\prod_{j=1}^{J} p(G_j^\varrho|G_0^\varrho, \boldsymbol{z}_j)$, the non-constant term in (8) with respect to $q(G_j|G_0, \boldsymbol{z}_j)$ is

$$\begin{aligned}\liminf_{\varrho} \Big\{ & \sum_{j=1}^{J} \mathrm{E}_{q(G_0^\varrho)}\mathrm{E}_{q(\boldsymbol{z}_j)}\mathrm{E}_{q(G_j^\varrho|G_0^\varrho,\boldsymbol{z}_j)} \log p(G_j^\varrho|G_0^\varrho, \boldsymbol{z}_j) \\ & - \log q(G_j^\varrho|G_0^\varrho, \boldsymbol{z}_j) \Big\}.\end{aligned}$$

Note that the above expression can be viewed as the negative of a KL divergence whose maximum is zero. Therefore, to enable the NPELBO to reach the maximum, the optimal conditional variational distribution for G_j should be $p(G_j|G_0, \boldsymbol{z}_j)$. Consequently, NPELBO in (8) does not contain any term related to $q(G_j|G_0, \boldsymbol{z}_j)$. In Appendix C.3, we derive NPELBO with respect to $q(G_0^\varrho)$ as

$$\begin{aligned}\liminf_{\varrho} \Big\{ & \sum_{j=1}^{J} \mathrm{E}_{q(G_0^\varrho)}\mathrm{E}_{q(\boldsymbol{z}_j)} \log \mathrm{E}_{p(G_j^\varrho|G_0^\varrho)} p(\boldsymbol{z}_j|G_j^\varrho) \\ & - \mathcal{KL}\big(q(G_0^\varrho)\|p(G_0^\varrho)\big) \Big\}\end{aligned} \tag{9}$$

up to a constant. It is important to note that $\mathrm{E}_{p(G_j^\varrho|G_0^\varrho)}$ is with respect to the prior $p(G_j^\varrho|G_0^\varrho)$ instead of the variational distribution $q(G_j^\varrho|G_0^\varrho)$, and hence this expectation can often be calculated analytically in HBNP models due to the conjugacy.

4.2 Empirical Distribution and Evidence Lower Bound

Within the conditional variational freamework, for the task of adaptive truncation, CATVI integrates Monte Carlo sampling to variational inference by iterating the following steps till convergence, (i) using Monte Carlo sampling to get an empirical optimal variational distribution for local variables $\boldsymbol{z}$ and (ii) updating the variational distributions for global variables G_0 and β.

Empirical distribution From the entire data $\boldsymbol{x}$, we randomly sample a subset $\{\boldsymbol{x}_s : \boldsymbol{x}_s \in \boldsymbol{x}\}_{s=1}^{S}$, where S is the batch size with $S \ll J$. Given a partition Ω in the current training iteration, we aim to update the parameters for $q(G_0^\varrho)$ conditional on $q(\beta)$ and $\{q(\boldsymbol{z}_s)\}_{s=1}^{S}$. While standard stochastic variational inference updates parameters analytically, we use Monte Carlo sampling to draw T_s samples for each $\boldsymbol{z}_s$ from $q(\boldsymbol{z}_s)$, thus constructing an empirical distribution,

$$\hat{q}(\boldsymbol{z}_s) = \frac{1}{T_s}\sum_{t=1}^{T_s} \delta_{\hat{\boldsymbol{z}}_{s,t}}, \quad \hat{\boldsymbol{z}}_{s,t} \sim q(\boldsymbol{z}_s).$$

Empirical evidence lower bound Using the empirical distribution $\hat{q}(\boldsymbol{z}_s)$, we obtain an empirical evidence lower bound with respect to $q(G_0^\varrho)$, $\widehat{\mathrm{ELBO}}^\varrho$, by replacing $q(\boldsymbol{z}_s)$ in (9) with $\hat{q}(\boldsymbol{z}_s)$, that is,

$$\begin{aligned}\sum_{s=1}^{S}\sum_{t=1}^{T_s} \frac{J}{ST_s} & \mathrm{E}_{q(G_0^\varrho)} \log \mathrm{E}_{p(G_s^\varrho|G_0^\varrho)} p(\hat{\boldsymbol{z}}_{s,t}|G_s^\varrho) \\ & - \mathcal{KL}\big(q(G_0^\varrho)\|p(G_0^\varrho)\big)\end{aligned} \tag{10}$$

up to a constant. It is obvious that $\mathrm{E}(\widehat{\mathrm{ELBO}}^\varrho) = \mathrm{ELBO}^\varrho$, thus satisfying the key condition for stochastic variational inference (Hoffman et al., 2013), that is, the random gradient is unbiased. Therefore, according to (10), we can use the random gradient generated from $\hat{\boldsymbol{z}}_s = \{\hat{\boldsymbol{z}}_{s,t}\}_{t=1}^{T_s}$ to update the parameters for $q(G_0^\varrho)$.

Resampling We next present the procedure to get the empirical distribution $\hat{q}(\boldsymbol{z}_s)$. As G_s is integrated out, the local latent variables $\{z_{si}\}_{i=1}^{N_s}$ can not be sampled independently when we use Monte Carlo sampling to draw $\hat{\boldsymbol{z}}_s$ given $q(G_0^\varrho)$ and $q(\beta)$. Therefore, we propose the following Gibbs sampling approach to get samples under optimal variational distributions. Conditional on $q(G_0^\varrho)$, $q(\beta)$ and samples $\hat{\boldsymbol{z}}_{s,i^-} = \{\hat{z}_{sl} : l = 1, \ldots, N_s, l \neq i\}$, it follows from (8) that the optimal variational distribution of $\log q(z_{si})$ is proportional to

$$\begin{aligned}& \mathrm{E}_{q(G_0^\varrho)}\mathrm{E}_{p(G_j^\varrho|G_0^\varrho)} p(z_{si}, \hat{\boldsymbol{z}}_{s,i^-}|G_s^\varrho) \\ & + \mathrm{E}_{q(\beta)} \log p(x_{si}|z_{si}, \beta).\end{aligned} \tag{11}$$

Then we sample $\hat{z}_{si} \sim q(z_{si})$ for each i iteratively, which constructs a Markov chain. Noting that $q(z_{si})$ is

often multinomial, sampling from its logarithm is commonly used. After the convergence, we can resample $\hat{\boldsymbol{z}}_{s,1}, \ldots, \hat{\boldsymbol{z}}_{s,T_s}$ from the stable Markov chain to update the parameters of $q(G_0^o)$ according to (10). Similarly, we derive the empirical evidence lower bound with respect to $q(\beta)$ in Appendix B.1 and can update the parameters for $q(\beta)$ using $\hat{\boldsymbol{z}}_{s,1}, \ldots, \hat{\boldsymbol{z}}_{s,T_s}$ correspondingly.

4.3 Adaptive Truncation

Finally, we seek to obtain the finite partition Ω that could reach the limit inferior in NPELBO. Rather than having Ω fixed on a universal truncation level, we enable the dimension of Ω to gradually adjust to a stable level. This partition or truncation is dependent on data-fitting and embedded within the optimization process, providing another key advantage of using a Monte Carlo sampling scheme in the stochastic variational inference framework.

Partition refinement According to the structure of HBNP models, $\hat{z}_{si}$ are sampled from the atomic support of G_0, $\{\phi_k\}_{k=1}^{\infty}$. Without loss of generality, we assume that the current partition Ω consists of atomic elements $\phi_1, \ldots, \phi_K \in \{\phi_k\}_{k=1}^{\infty}$ and their complement $\phi_0 = \Omega / \{\phi_1, \ldots, \phi_K\}$. Under this partition, $q(z_{si} \in \phi_0)$ is positive given (11), and hence $\hat{z}_{si}$ can be sampled within ϕ_0, that is, $\hat{z}_{si}$ is a new sample, distinct from $\phi_1, \ldots, \phi_K$. If this happens, we draw a new ϕ_{K+1} and refine the partition as $\big(\phi_0, \phi_1, \ldots, \phi_K, \phi_{K+1}\big)$, where ϕ_0 is updated as $\Omega / \{\phi_1, \ldots, \phi_K, \phi_{K+1}\}$.

Remark: The partition refinement procedure reaches the limit inferior of empirical evidence lower bound as follows. Since there is no sampling within ϕ_0 after each update, to minimize the KL divergence, the posterior should be proportional to the prior on ϕ_0, $q\big(G_0(\phi_0)\big) \propto p\big(G_0(\phi_0)\big)$. Moreover, if we further partition ϕ_0 into $\phi_0^1 \cup \phi_0^2$, the KL divergence stays the same. Thus, $\mathrm{E}(\widehat{\mathrm{ELBO}}^{\Omega}) = \mathrm{ELBO}^{\Omega} = \mathrm{NPELBO}$. See Appendix C.5 for a justification.

We summarize the CATVI algorithm in Algorithm 1.

Algorithm 1: CATVI Algorithm

```
Initialize the partition Ω, the parameters for
q(G_0), q(β) and set up the step-size {ρ_τ}_{τ≥1}.
repeat
  Randomly select x_1, ..., x_S from the dataset.
  for s ∈ {1, ..., S} do
    repeat
      for i ∈ {1, ..., N_s} do
        Sample ẑ_si | q(G_0), q(β), ẑ_{s,i^-} .
        if Sampling a new ẑ_si then
          Refine the partition Ω.
        end if
      end for
    until convergence
    Resample ẑ_s = {ẑ_{s,t}}_{t=1}^{T_s}.
  end for
  Update parameters for q(G_0) and q(β) given
  samples {ẑ_s}_{s=1}^{S} using the step-size ρ_τ .
until convergence
```

5 APPLICATIONS IN TOPIC MODELS

5.1 CATVI for the HDP Model

We apply the proposed CATVI method to the HDP model. Specifically, we factorize the variational distributions in the conditional setting and specify the variational family as follows. First, the variational distribution of G_s for each s is given by $q(G_s | G_0, \boldsymbol{z}_s) = \mathrm{DP}\big(\sum_{k=1}^{\infty} n_{sk}\delta_{\phi_k} + G_0\big)$, where $n_{sk} = \sum_{i=1}^{N_s} I(z_{si} = \phi_k)$ and $I(\cdot)$ is the indicator function. Second, $q(\beta_k)$ for each topic k is set as a W-dimensional Dirichlet distribution, $q(\beta_k) = \mathrm{Dirichlet}(\lambda_k)$, where $\lambda_k = (\lambda_{k1}, \ldots, \lambda_{kW})^{\mathrm{T}}$ is the parameter of vocabulary distribution for topic k. The variational distribution for topics without any observation remains the same as the prior, hence $q(\beta_0) = \mathrm{Dirichlet}(\eta)$. Third, we specify the variational family for G_0 using spike and slab distributions (Andersen et al., 2017) as $q(G_0) = \sum_{k=1}^{K} m_k \delta_{\phi_k} + m_0 \mathrm{DP}(\alpha H)$, such that $\sum_{k=0}^{K} m_k = 1$. Finally, following (11) we use Monte Carlo sampling to obtain samples $\{\hat{\boldsymbol{z}}_s\}_{s=1}^{S}$, avoiding the need to parametrize their variational distributions.

As different samples in $\{\hat{\boldsymbol{z}}_s\}_{s=1}^{S}$ are used to represent different topic clusters in topic modelling, their exact values in the sample space do not contain any statistical information. We can then simply index the topics from 1 to K and denote the different clusters by distinct points $\phi_1, \ldots, \phi_K$ in Ω, and cluster 0 is the topic without any observation. Given samples $\{\hat{\boldsymbol{z}}_s\}_{s=1}^{S}$, we define the number of topics with observations by $K = \sum_{k=0}^{\infty} I(\sum_{s=1}^{S}\sum_{t=1}^{T_s} \hat{n}_{sk,t} > 0)$, where $\hat{n}_{sk,t} = \sum_{i=1}^{N_s} I(\hat{z}_{si,t} = \phi_k)$. In Appendix C.4, we rely on (10) to derive the empirical evidence lower bound with respect to $q(G_0)$,

$$\begin{aligned} &\alpha \log m_0 - \sum_{k=0}^{K} \log m_k \\ &+ \sum_{s=1}^{S}\sum_{k=1}^{K}\sum_{t=1}^{T_s} \frac{J}{ST_s} \log \frac{\Gamma(\gamma m_k + \hat{n}_{sk,t})}{\Gamma(\gamma m_k)} \end{aligned} \tag{12}$$

up to a constant. According to Algorithm 1, we repeatedly select documents of a batch size S, sample $\{\hat{\boldsymbol{z}}_s\}_{s=1}^S$, and update parameters for G_0 and β iteratively until the empirical evidence lower bound converges to its maximum. During Gibbs sampling, once a document is sampled in cluster 0, we add a new cluster $K+1$, thus partitioning Ω to be $(K+1)$-dimensional, with K single points $\{\phi_k\}_{k=1}^K$ and one complement set $\phi_0 = \Omega/\{\phi_k\}_{k=1}^K$. During the training, this procedure is repeated until Ω is optimized. See Appendix B.2 for detailed steps on updating the variational parameters and refining the partition.

5.2 CATVI for Generic HBNP Models

The CATVI algorithm can also be applied to a general class of HBNP models, where the global prior G_0 is generated from a completely random measure. In these models, the concentration parameter for any G_j is not fixed, and G_0 is not restricted to be a probability measure. The corresponding inference algorithm is similar to that of the HDP model, but requires a new parameter μ to approximate $G_0(\Omega)$. We choose the variational family for the global prior G_0 as $q(G_0) = \mu\big(\sum_{k=1}^K m_k\delta_{\phi_k} + m_0\widetilde{N}(\alpha H)\big)$, where $\widetilde{N}$ is the normalization of the corresponding completely random measure and $\sum_{k=0}^K m_k = 1$. We provide the corresponding empirical evidence lower bounds and algorithms to infer more general HBNP models, including ΓDP model, in Appendix B.3.

6 RELATIONSHIP TO RELATED WORKS

In this section, we discuss several advantages of CATVI compared with traditional methods (Hoffman et al., 2013; Wang et al., 2011; Wang and Blei, 2012), although these are specific to the inference for the HDP model. First, CATVI replaces the unrealistic mean-field assumption with the conditional setting to capture the correlation structure among latent variables. Second, CATVI approximates the posterior group-wisely instead of updating the stick-breaking parameters sequentially, and hence avoids the gradient vanishing problem. By contrast, Hoffman et al. (2013) and Wang et al. (2011) perform inference separately over each atomic location and weight of G_0 using the stick-breaking representation $G_{0K} = g_{0K}\prod_{k=1}^{K-1}(1-g_{0k})$, where g_{0k}s are the representation parameters. However, this may cause the gradient vanishing problem of G_{0K} if k is large, because $\prod_{k=1}^{K-1}(1-g_{0k})$ is close to zero. Third, these traditional methods universally truncate the dimension of G_0 to a fixed level, contradicting the motivation and advantages of using HBNP models. Finally, CATVI is guaranteed to maximize the NPELBO. By comparison, Wang and Blei (2012) update parameters using the locally collapsed Gibbs sampling, but their work leads to an approximation that fails to maximize the ELBO, especially when the variance of distributions is large.

From a computational perspective, CATVI inherits the fast speed of stochastic variational inference, while other methods that truncate the dimension in a truly nonparametric way are very slow, such as the split-merge variational inference (Bryant and Sudderth, 2012) and the pure Gibbs sampling (Teh et al., 2006). To check the split-merge criterion, the split-merge variational inference requires calculating the likelihood before and after a split or merge, which is computationally infeasible in practice. Moreover, the pure Gibbs sampling is not scalable as well. As pure Gibbs sampling does not have batch selection, the Markov chains would converge very slowly when the sample size is large. As a result, these methods cannot be used to handle big data.

7 EXPERIMENTS

7.1 Datasets and Architectures

We apply the CATVI algorithm to three large datasets, *arXiv*, *NYT* and *Wiki*, and compare the performance of CATVI with the online variational inference (OVI) (Wang et al., 2011), the memorized online variational inference (MOVI) (Hughes and Sudderth, 2013), the split-merge variational inference (SMVI) (Bryant and Sudderth, 2012) and Gibbs sampling (GS) (Teh et al., 2004).

arXiv The corpus contains descriptive metadata of articles on *arXiv* up to September 1, 2019, resulting in 1.03M documents and 44M words from a vocabulary of 7,500 terms.

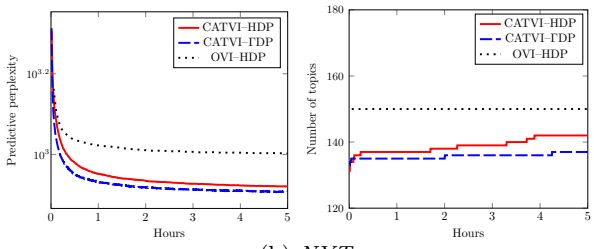
NYT The corpus contains all articles published by *New York Times* from January 1987 to June 2007 (Sandhaus, 2008), resulting in 1.56M documents and 176M words from a vocabulary of 7,600 terms.

Wiki The corpus contains entries from all English *Wikipedia* websites on January 1, 2019, resulting in 4.03M documents and 423M words from a vocabulary of 8,000 terms.

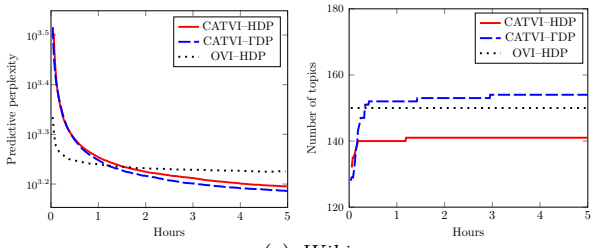
For the preprocessing, stemming and lemmatization are used to clean the raw text, and then words with too high or too low frequency, as well as common stop words, are filtered out.

To evaluate the performance of CATVI, we set aside a test set of 10,000 documents for each dataset and

(a) *arXiv*



(b) *NYT*



(c) *Wiki*

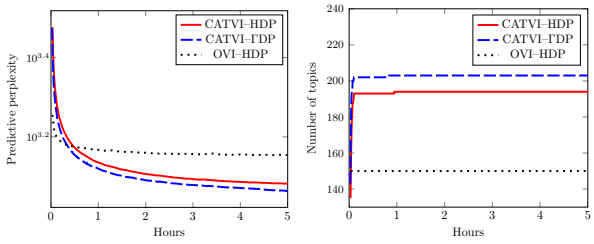


Figure 2: Left column: plots for the perplexity vs the running time up to 5 hours, Right column: plots for the number of topics vs the running time.

calculate the predictive perplexity as

$$\text{perplexity} = \exp\left\{-\frac{\sum_{j\in \mathrm{D}_{\text{test}}} \log p(\boldsymbol{x}_j^{\text{test}}|\boldsymbol{x}_j^{\text{train}}, \mathrm{D}_{\text{train}})}{\sum_{j\in \mathrm{D}_{\text{test}}} |\boldsymbol{x}_j^{\text{test}}|}\right\},$$

where $\mathrm{D}_{\text{train}}$ and D_{test} represent the training and test data, respectively, $\boldsymbol{x}_j^{\text{train}}$ and $\boldsymbol{x}_j^{\text{test}}$ are the training and test words in test document j, respectively, and $|\boldsymbol{x}_j^{\text{test}}|$ is the number of words in $\boldsymbol{x}_j^{\text{test}}$ (Ranganath and Blei, 2018). The perplexity measures the uncertainty of fitted models, where a lower perplexity will result in a better language model with higher predictive likelihood. Since the perplexity can not be computed exactly, the standard routine uses $\mathrm{D}_{\text{train}}$ to compute the variational distribution for β and G_0, then obtains the variational distribution for G_j based on G_0 and $\boldsymbol{x}_j^{\text{test}}$, and then approximates the likelihood by $p(\boldsymbol{x}_j^{\text{test}}|\boldsymbol{x}_j^{\text{train}}) = \prod_{w\in\boldsymbol{x}_j^{\text{test}}} \sum_{k=0}^{K} \overline{G}_{jk}\overline{\beta}_{kw}$, where $\overline{G}_{jk}$ and $\overline{\beta}_{kw}$ are the variational expectations of G_{jk} and β_{kw}, respectively (Blei et al., 2003). Experiments are run with the three datasets above using both the HDP and ΓDP models. For the HDP model, we set the hyperparameters as $\alpha = \gamma = \eta = 5$, where α and γ are the concentration parameters for G_0 and G_j respectively, and η is the hyperparameter for the prior on the distribution of words. The initial number of topics is set to be 100. The parameters are then optimized using stochastic gradient descent, with a batch size of 256 and a linear decaying learning rate adopted in Hoffman et al. (2010). For the ΓDP model, we use the same settings but discard γ. In the experiments, we remove clusters with fewer than 1 document during the training.

Table 1: A summary of predictive perplexity results.

Model	Method	*arXiv*	*NYT*	*Wiki*
HDP	GS	3175	2635	1807
HDP	MOVI	1901	2921	1876
HDP	SMVI	1917	2866	1877
HDP	OVI	1005	1681	1422
HDP	CATVI	832	1569	1207
ΓDP	CATVI	808	1536	1157

7.2 Empirical Results

Predictive perplexity The top row of Figure 2 plots the predictive perplexity as a function of running time for the three comparison methods using the three datasets. As MOVI, SMVI and GS provide much higher perplexities, we do not plot their results in Figure 2. Table 1 reports numerical summaries for all comparison methods. In particular, as GS can not scale to large datasets, we use a subset with 500 documents to run the experiments. Several conclusions can be drawn here. First, on all three datasets, CATVI uniformly outperforms competing methods. The improvement is highly consequential, especially for *arXiv* and *Wiki*. For *NYT*, there is moderate improvement, likely due to the long length of documents in this corpus. Second, for each dataset, the ΓDP model attains a lower perplexity than the HDP model, consistent with the fact that the ΓDP model removes a restriction of the HDP model and hence is more flexible. Third, CATVI is empirically shown to be computationally efficient, reaching the lowest perplexity within the same training time. Although it involves Monte Carlo sampling, the perplexity converges fast. This is because the convergence of local Markov chains to assign words to topics is accelerated by a clear topic-words clustering as the global variational distributions approach to the optimal.

Number of topics The bottom row of Figure 2 plots the number of topics during the training process. For OVI, the number of topics remains constant at the prespecified value, while for CATVI, this value first increases steeply and then converges to a stable level.

For example, the number of topics in *Wiki* sharply increase from 100 to around 190 for the HDP model and around 200 for ΓDP model. The sharp increase is driven by the data complexity, while the stable level is achieved due to the dimension penalty effect from the priors in HBNP models. Although the estimation of the number of topics is not consistent, CATVI can provide some useful information about topics in data. For instance, the data from the *arXiv* corpus in these experiments are limited to abstracts of scientific articles, and thus it has the smallest number of topics. By contrast, *NYT* is a compilation of all new articles covering a wider range of areas, and hence consists of more topics. Similarly, *Wiki* has the largest number of topics as it contains almost every aspect of an encyclopedia. It is important to note that we do not need to set a fixed number of topics before the inference. Instead, CATVI starts from an initial value, for example 100 in our experiments, then automatically converges to a stable optimal number of topics.

Topic-words clustering CATVI is shown to reveal much better linguistic results. To compare CATVI with OVI for the HDP model, we report the top 12 words in the top 10 topics with biggest weights for both methods on *arXiv* and *Wiki* in Tables 2a and 2b, respectively. We observe a few apparent patterns. First, the topic-word clusters from CATVI hardly contains replicated topics, whereas those from OVI results have similar word components, such as those shown in blue in columns 1-6 in the bottom part of Table 2a. An ideal topic-word clustering should allocate these words into just one topic. However, the prespecified number of topics is fixed at 150 in OVI, which is larger than the ground truth, resulting in generating replicated topics. By contrast, the topic-word clustering by CATVI does not have such redundancy. It is apparent that our top 10 topics are mostly distinct. Second, CATVI leads to much clearer topic-word clustering. For both datasets, our results indicate that all of our detected words within any column are highly relevant and should intuitively be grouped into one cluster with clear linguistic meaning. For example, column 7 of Table 2b for CATVI presents several words all related to military, but words in the same column for OVI seem to be a mixture of several loosely connected topics including 'human, character, reveal, episode, comic, voice', 'human, earth' and 'human, kill, attack, fight, battle, doctor'. This mixture of topics makes the topic-word clustering in this column ambiguous. Furthermore, CATVI identifies a topic about popular English given names in column 5 of Table 2b. Although these given names are not shown in a single document, CATVI can successfully discover that they belong to one topic, while OVI fails. This is because

Table 2: Top 12 words in top 10 topics.

(a) *arXiv*

	1	2	3	4	5	6	7	8	9	10
CATVI	galaxy	group	network	neutrino	star	gaug	prove	algorithm	collision	test
	cluster	algebra	learn	higg	dwarf	string	bound	optim	product	error
	redshift	construct	train	matter	survey	brane	theorem	converge	decay	samples
	samples	finit	neural	dark	object	symmetry	finit	solve	hadron	statist
	luminos	lie	image	decay	binari	dimension	class	linear	jet	uncertainties
	formation	categories	deep	standard	variable	couple	position	approximate	transverse	fit
	survey	prove	dataset	couple	cluster	action	inequalities	gradient	gev	systematic
	agnes	class	feature	mix	stellar	conform	dimension	minim	lhc	accuracy
	lar	complex	task	boson	period	construct	converge	matrix	cross	correct
	star	map	object	lepton	photometr	correspond	continual	iter	section	procedure
	populated	invariable	convolut	violate	distanc	dual	regular	spars	quark	improve
	host	manifold	detect	symmetry	samples	background	compact	constraint	collid	bias
OVI	galaxy	galaxy	star	xray	emiss	galaxy	higg	star	emiss	radio
	cluster	redshift	cluster	emiss	star	line	neutrino	planet	gammaray	emiss
	halo	source	abundance	source	region	emiss	decay	period	source	galaxy
	star	survey	galaxy	accret	line	gas	dark	orbit	grb	source
	stellar	samples	stellar	kev	gas	star	boson	dwarf	xray	xray
	formation	cluster	metal	line	dust	redshift	matter	binari	ray	jet
	velocity	luminos	age	variable	disk	absorption	standard	detect	detect	line
	dark	agnes	populated	spectral	molecular	samples	couple	stellar	burst	region
	gas	xray	ngc	star	cloud	quasar	gev	transit	flux	cluster
	matter	radio	dwarf	flux	detect	luminos	particle	variable	radio	detect
	profile	star	samples	spectrum	formation	region	mix	light	spectrum	gas
	disk	optic	giant	detect	galaxy	detect	symmetry	companion	jet	star

(b) *Wiki*

	1	2	3	4	5	6	7	8	9	10
CATVI	tell	increase	band	human	james	claim	armies	process	polit	album
	tried	effect	album	natur	robert	issue	battle	model	parti	chart
	want	case	guitar	tradition	charles	announce	attack	inform	union	song
	friend	process	vocal	term	david	critic	troop	effect	communist	track
	leave	measure	track	idea	thomas	controversi	command	problem	movement	video
	ask	caus	rock	view	richard	proposal	soldier	experience	independence	billboard
	feel	require	drum	word	michael	agreement	military	test	social	label
	decide	rate	bass	theorie	frank	polit	fight	example	republic	peak
	turn	example	song	philosophies	peter	allegation	tanks	research	leader	week
	good	reduce	tour	culture	andrew	statement	brigade	specific	worker	digitated
	away	possibilities	studio	believe	brown	agree	german	individual	socialist	hot
	believe	occur	label	conception	henry	minister	capture	object	liberal	remix
OVI	album	episode	actor	album	album	episode	character	novel	animal	ship
	band	tell	movi	song	song	televis	kill	character	episode	navies
	song	character	character	band	chart	drama	human	love	character	class
	track	kill	critic	tour	video	actor	earth	poem	voice	boat
	guitar	friend	cast	love	track	comedies	attack	london	movi	naval
	vocal	leave	review	blue	billboard	actress	reveal	king	video	command
	rock	tried	televis	artist	love	movi	episode	tell	air	vessel
	tour	relationship	episode	rock	label	theatre	comic	fiction	dvd	submarine
	chart	need	scene	track	version	voice	fight	narrated	televis	gun
	studio	love	theatre	label	week	uncredit	doctor	friend	ray	fleet
	bass	reveal	picture	chart	peak	nominal	battle	mother	blu	sail
	drum	mother	love	singer	remix	cast	voice	critic	song	destroy

CATVI does not force the topics to merge together if the prespecified number of topics is not large enough, thus reducing the noise in the clusters.

We also perform sensitivity analysis of CATVI using *arXiv* under the HDP model as an example. The left and right panels of Figure 3 in Appendix E respectively plot the results as the batch size varies from 128 to 1024 and the initial number of topics varies from 60 to 140. We observe that the performance is not sensitive to the change of these hyperparameters. Moreover, the best results are obtained for the case with a smaller batch size and a larger initial number of topics.

8 DISCUSSION

CATVI can also be applied to other HBNP models including, for example, hierarchical Pitman–Yor process model (Teh and Jordan, 2010) and hierarchical beta process model (Thibaux and Jordan, 2007). CATVI will provide more advantages in these applications, because the hierarchical Pitman–Yor process, with heavy tail behavior, and the hierarchical beta process, with sparse structure, may suffer more from the universal truncation.

Acknowledgments

We thank the anonymous reviewers for useful comments during the review process.

Opinions expressed in this paper are those of the authors, and do not necessarily reflect the view of J.P. Morgan. Opinions and estimates constitute our judgement as of the date of this Material, are for informational purposes only and are subject to change without notice. This Material is not the product of J.P. Morgan's Research Department and therefore, has not been prepared in accordance with legal requirements to promote the independence of research, including but not limited to, the prohibition on the dealing ahead of the dissemination of investment research. This Material is not intended as research, a recommendation, advice, offer or solicitation for the purchase or sale of any financial product or service, or to be used in any way for evaluating the merits of participating in any transaction. It is not a research report and is not intended as such. Past performance is not indicative of future results. Please consult your own advisors regarding legal, tax, accounting or any other aspects including suitability implications for your particular circumstances. J.P. Morgan disclaims any responsibility or liability whatsoever for the quality, accuracy or completeness of the information herein, and for any reliance on, or use of this material in any way. Important disclosures at: www.jpmorgan.com/disclosures.

References

Andersen, M. R., Vehtari, A., Winther, O., and Hansen, L. K. (2017). Bayesian inference for spatio-temporal spike-and-slab priors. *Journal of Machine Learning Research*, 18(1):5076–5133.

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

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Bryant, M. and Sudderth, E. B. (2012). Truly nonparametric online variational inference for hierarchical Dirichlet processes. In *Advances in Neural Information Processing Systems 25*, pages 2699–2707.

Caron, F. and Fox, E. B. (2017). Sparse graphs using exchangeable random measures. *Journal of the Royal Statistical Society: Series B*, 79(5):1295–1366.

Ghosal, S. and Van der Vaart, A. (2017). *Fundamentals of Nonparametric Bayesian Inference*. Cambridge University Press, Cambridge.

Hoffman, M., Bach, F. R., and Blei, D. M. (2010). Online learning for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems 23*, pages 856–864.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347.

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

Kingman, J. F. C. (1993). *Poisson Processes*. Clarendon Press, Oxford.

Matthews, A. G. d. G., Hensman, J., Turner, R., and Ghahramani, Z. (2016). On sparse variational methods and the Kullback-Leibler divergence between stochastic processes. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*.

Ranganath, R. and Blei, D. M. (2018). Correlated random measures. *Journal of the American statistical Association*, 113(521):417–430.

Roychowdhury, A. and Kulis, B. (2015). Gamma processes, stick-breaking, and variational inference. In *Proceedings of the 8th International Conference on Artificial Intelligence and Statistics*.

Sandhaus, E. (2008). *The New York Times annotated corpus*. Linguistic Data Consortium, Philadelphia.

Sudderth, E. B. and Jordan, M. I. (2009). Shared segmentation of natural scenes using dependent Pitman–Yor processes. In *Advances in Neural Information Processing Systems 21*, pages 1585–1592.

Teh, Y., Jordan, M., Beal, M., and Blei, D. (2004). Sharing Clusters among Related Groups: Hierarchical Dirichlet Processes. In *Advances in Neural Information Processing Systems 17*.

Teh, Y. W. and Jordan, M. I. (2010). Hierarchical Bayesian nonparametric models with applications. In *Bayesian Nonparametrics*, pages 158–207. Cambridge University Press.

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

Teh, Y. W., Kurihara, K., and Welling, M. (2008). Collapsed variational inference for HDP. In *Advances in Neural Information Processing Systems 20*.

Thibaux, R. and Jordan, M. I. (2007). Hierarchical beta processes and the Indian buffet process. In *Proceedings of the 11th International Conference on*

Artificial Intelligence and Statistics, volume 2, pages 564–571.

Wang, C. and Blei, D. M. (2012). Truncation-free online variational inference for Bayesian nonparametric models. In *Advances in Neural Information Processing Systems 25*.

Wang, C., Paisley, J., and Blei, D. M. (2011). Online variational inference for the hierarchical Dirichlet process. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*.

Williamson, S. A. (2016). Nonparametric network models for link prediction. *Journal of Machine Learning Research*, 17(202):1–21.

Xu, K., Srivastava, A., and Sutton, C. (2019). Variational Russian Roulette for deep Bayesian nonparametrics. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6963–6972.

Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, N., and Khazaeni, Y. (2019). Bayesian nonparametric federated learning of neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 7252–7261.

Supplementary Material: CATVI: Conditional and Adaptively Truncated Variational Inference for Hierarchical Bayesian Nonparametric Models

This supplementary material contains a short review of completely random measures in Appendix A, CATVI algorithm and its applications to the HDP model and the ΓDP model in Appendix B, technical proofs and derivations in Appendix C, computational complexity analysis and code in Appendix D and sensitivity analysis results in Appendix E.

A A Short Review of Completely Random Measures

Suppose that $(\Omega, \mathcal{F})$ is a Polish sample space, Θ is the set of all bounded measures on $(\Omega, \mathcal{F})$ and $\mathcal{M}$ is a σ-algebra on Θ. A random measure G on $(\Omega, \mathcal{F})$ is a transition kernel from $(\Theta, \mathcal{M})$ into $(\Omega, \mathcal{F})$ such that (i) $G \mapsto G(A)$ is $\mathcal{M}$-measurable for any $A \in \mathcal{F}$ and (ii) $A \mapsto G(A)$ is a measure for any realization of G (Ghosal and Van der Vaart, 2017). For example, a Dirichlet process P with base measure P_0 satisfies

$$\big(P(A_1), \ldots, P(A_n)\big) \sim \text{Dirichlet}\big(P_0(A_1), \ldots, P_0(A_n)\big)$$

for any partition $\Omega = (A_1, \ldots, A_n)$ of Ω, that is, a finite number of measurable, nonempty and disjoint sets such that $\bigcup_{i=1}^n A_i = \Omega$. The Dirichlet process is denoted by $P \sim \text{DP}(P_0)$ or $P \sim \text{DP}(\alpha H)$ with concentration parameter $\alpha = P_0(\Omega)$ and center measure $H = \alpha^{-1} P_0$. Moreover, a random measure is called a completely random measure (Kingman, 1993) if it also satisfies the condition that (iii) $P(A_i)$ is independent of $P(A_j)$ for any disjoint subsets A_i and A_j in Ω. Completely random measures and their normalizations (Ghosal and Van der Vaart, 2017), for example, the Gamma process and Dirichlet process, respectively, are commonly used as priors for infinite-dimensional latent variables in HBNP models, because their realizations are atomic measures with countable-dimensional supports.

A completely random measure (Kingman, 1993) is characterized by its Laplace transform,

$$\text{E}\big[e^{-tP(A)}\big] = \exp\Big\{ -\int_A \int_{(0,\infty]} (1 - e^{-t\pi}) v^c(dx, ds) \Big\},$$

where A is any measurable subset of Ω and $v^c(dx, ds)$ is called the Lévy measure. If $v^c(dx, ds) = \kappa(dx) v(ds)$, where $\kappa(\cdot)$ and $v(\cdot)$ are measures on Ω and $(0, \infty]$, respectively, the completely random measure is homogeneous (Ghosal and Van der Vaart, 2017). In such a case, we call $v(\cdot)$ the weight intensity measure. We can view completely random measure as a Poisson process on the product space $\Omega \times (0, \infty]$ using its Lévy measure as the mean measure.

B CATVI Algorithm

B.1 Empirical ELBO for $q(\beta)$ and $q(z_{si})$

To maximize the NPELBO, we iterate the following three steps: (i) randomly select a small batch from the entire data, (ii) sample $\{\hat{\boldsymbol{z}}_s\}_{s=1}^S$ by Monte Carlo method, and (iii) update $q(G_0^a)$ and $q(\beta)$ in the stochastic variational inference framework.

In an analogy to (9), the NPELBO with respect to $q(\beta)$ is

$$\liminf_a \Big\{ \sum_{j=1}^J \text{E}_{q(\beta)} \text{E}_{q(\boldsymbol{z}_j)} \log p(\boldsymbol{x}_j | \boldsymbol{z}_j, \beta) - \mathcal{KL}\big(q(\beta) \| p(\beta)\big) \Big\}, \tag{A.1}$$

and the empirical evidence lower bound with respect to $q(\beta)$, $\widehat{\text{ELBO}}^{\Omega}$, is,

$$\sum_{s=1}^{S}\sum_{t=1}^{T_s}\frac{J}{ST_s}\mathrm{E}_{q(\beta)}\log p(\boldsymbol{x}_s|\hat{\boldsymbol{z}}_{s,t},\beta)-\mathcal{KL}\big(q(\beta)|p(\beta)\big) \tag{A.2}$$

up to a constant and then we can update its parameter with the corresponding random gradient in a similar way. Moreover, the NPELBO with respect to $q(z_{si})$ is

$$\liminf_{\Omega}\Big\{\mathrm{E}_{q(G_0^{\Omega})}\mathrm{E}_{q(\boldsymbol{z}_j)}\log\mathrm{E}_{p(G_j^{\Omega}|G_0^{\Omega})}p(\boldsymbol{z}_j|G_j^{\Omega})+\mathrm{E}_{q(\boldsymbol{z}_j)}\mathrm{E}_{q(\beta)}\log p(\boldsymbol{x}_j|\boldsymbol{z}_j,\beta)-\mathrm{E}_{q(\boldsymbol{z}_j)}\log q(\boldsymbol{z}_j)\Big\}. \tag{A.3}$$

Factorizing $\mathrm{E}_{q(\boldsymbol{z}_j)}$ as $\mathrm{E}_{q(\boldsymbol{z}_{ji})}\mathrm{E}_{q(\boldsymbol{z}_{j,i^-})}$ leads to (11). We summarize the details of CATVI algorithm in Algorithm 1.

B.2 CATVI for the HDP Model

We repeatedly select documents of a batch size and update parameters iteratively according to the following three steps, until the NPELBO attains its maximum.

Inference for G_0. There is no closed-form expression for the parameters $\{m_k\}_{k=0}^K$ to attain the maximum in (12). Moreover, the standard gradient descent algorithm fails in this case, because $\{m_k\}_{k=0}^K$ may easily exceed the simplex during the updating procedure. Instead, given the parameters $\{m_k^{(\tau)}\}_{k=0}^K$ in the τ-th iteration, we first define

$$m_k^*\propto\begin{cases}JS^{-1}\gamma\sum_{s=1}^{S}\big\{T_s^{-1}\sum_{t=1}^{T_s}\Phi(\gamma m_k^{(\tau)}+\hat{n}_{sk,t})-\Phi(\gamma m_k^{(\tau)})\big\}m_k^{(\tau)}-1 & k=1,\dots,K,\\ \alpha-1 & k=0,\end{cases} \tag{B.1}$$

where $\Phi(\cdot)$ denotes the log-gamma function, such that $\sum_{k=0}^K m_k^*=1$, and then we update the parameters by $m^{(\tau+1)}=(1-\rho_t)m^{(\tau)}+\rho_\tau m_k^*$, where ρ_t is the step size defined in Algorithm 1. This updating algorithm is consistent to the gradient descent after the inverse logit transformation. See Appendix C.6 for a justification. In the process of updating, the condition $\sum_{k=0}^K m_k^*=1$ always holds, and hence we eliminate the risk of exceeding the simplex.

Inference for β. By (A.2), we update the parameters for $q(\beta)$ using samples $\{\hat{\boldsymbol{z}}_s\}_{s=1}^S$. We define λ_{kw}^* for topic k and word w as,

$$\lambda_{kw}^*=\eta+\sum_{s=1}^{S}\sum_{t=1}^{T_s}\sum_{i=1}^{N_s}\frac{J}{ST_S}I(\hat{z}_{si,t}=\phi_k,x_{si}=w), \tag{B.2}$$

and update the parameter λ_k by $\lambda_k^{(\tau+1)}=(1-\rho_t)\lambda_k^{(\tau)}+\rho_\tau\lambda_k^*$ for each k, where $\lambda_k^*=(\lambda_{k1}^*,\dots,\lambda_{kW}^*)^{\mathrm{T}}$.

Sampling for $\boldsymbol{z}$. According to (11) we sample $\hat{z}_{si}$ conditional on $q(G_0)$ and $\hat{\boldsymbol{z}}_{si^-}$ by

$$q(z_{si}=\phi_k)\ \propto\ \begin{cases}(\gamma m_k+\hat{n}_{s,i^-}^k)\exp\big(\Phi(\lambda_{kx_{si}})-\Phi(\sum_{w=1}^W\lambda_{kw})\big) & k=1,\dots,K,\\ \gamma m_0\exp\big(\Phi(\eta)-\Phi(W\eta)\big) & k=0,\end{cases} \tag{B.3}$$

to construct the Markov chain, where $\hat{n}_{s,i^-}^k=\sum_{1\le l\le N_s,l\neq i}I(\hat{z}_{sl}=\phi_k)$. Whenever the sampled $\hat{z}_{si}$ is in ϕ_0, meaning $\hat{z}_{si}$ forms a new point not belonging to $\{\phi_1,\dots,\phi_K\}$, we need to update the partition and add a new topic indicated by ϕ_{K+1}. Otherwise the partition dimension remains the same. Iterating the sampling scheme till convergence, we obtain the samples $\{\hat{z}_{si,t}\}_{1\le s\le S,1\le i\le N_s,1\le t\le T_s}$ and corresponding $\{\hat{n}_{sk,t}\}_{1\le s\le S,1\le k\le K,1\le t\le T_s}$ for the selected chunk.

B.3 CATVI for the ΓDP Model

ΓDP releases the constraint of fixed concentration parameter γ in HDP. Therefore, the CATVI algorithm for ΓDP inherits the steps in (B.1) and (B.3), except that a parameter μ replaces the concentration parameter γ in both formulas.

We derive the empirical evidence lower bound in Appendix C.7 with respect to $q(G_0)$ as,

$$\sum_{k=1}^{K}\log v(\mu m_k)+\log u(\mu m_0)+\sum_{s=1}^{S}\frac{J}{S}\log\frac{\Gamma(\mu)}{\Gamma(\mu+N_s)}+\sum_{s=1}^{S}\sum_{k=1}^{K}\sum_{t=1}^{T_s}\frac{J}{ST_s}\log\frac{\Gamma(\mu m_k+\hat{n}_{sk,t})}{\Gamma(\mu m_k)}+K\log\mu \tag{B.4}$$

up to a constant, where $v(\cdot)$ is the weight intensity measure (see Appendix A) for the completely random measure, and $u(\cdot)$ is the density function for $G_0(\Omega)$ that can be derived using its Laplace transform. Therefore, we can update $\{m_k\}_{k=0}^K$ in the same way as the HDP model.

Similar to (B.1), we update $\{m_k\}_{k=0}^K$ according to

$$m_k^*\propto\begin{cases}JS^{-1}\mu\sum_{s=1}^{S}\big\{T_s^{-1}\sum_{t=1}^{T_s}\Phi(\mu m_k^{(\tau)}+\hat{n}_{sk,t})-\Phi(\mu m_k^{(\tau)})\big\}m_k^{(\tau)}-1 & k=1,\dots,K,\\ \alpha-1 & k=0,\end{cases} \tag{B.5}$$

and $m^{(\tau+1)}=(1-\rho_t)m^{(\tau)}+\rho_\tau m_k^*$. Moreover, in an analogy to (B.3), the probability to sample $\hat{z}_{si}$ is defined as

$$q(z_{si}=\phi_k)\ \propto\ \begin{cases}(\mu m_k+\hat{n}_{s,i^-}^k)\exp\big(\Phi(\lambda_{kx_{si}})-\Phi(\sum_{w=1}^{W}\lambda_{kw})\big) & k=1,\dots,K,\\ \mu m_0\exp\big(\Phi(\eta)-\Phi(W\eta)\big) & k=0.\end{cases} \tag{B.6}$$

Finally, we apply the gradient ascent to update μ. In Appendix C.7, we derive the gradient of empirical evidence lower bound with respect to μ as

$$g'(\mu)=\frac{\alpha-1}{\mu}-1+\frac{J}{S}\sum_{s=1}^{S}\Big\{\Phi(\mu)-\Phi(\mu+N_s)+\sum_{k=1}^{K}\frac{1}{T_s}\sum_{t=1}^{T_s}m_k\big(\Phi(\mu m_k+\hat{n}_{sk,t})-\Phi(\mu m_k)\big)\Big\}, \tag{B.7}$$

and then update μ by $\mu^{(\tau+1)}=\mu^{(\tau)}+\rho_\tau g'(\mu^{(\tau)})$.

C Technical Proofs and Derivations

C.1 Proof for (4)

By definition of induced measure, $q^\varrho(d\Theta)=Q(d\Theta)$ for any $\mathcal{M}$-measurable $d\Theta$, we have

$$\int_\Theta\log\frac{dq^\varrho}{dp^\varrho}dq^\varrho=\int_\Theta\log\frac{dq^\varrho}{dp^\varrho}dQ.$$

It follows from $\limsup_\varrho dq^\varrho/dp^\varrho=dQ/dP$ and the monotone convergence theorem that

$$\limsup_\varrho\int_\Theta\log\frac{dq^\varrho}{dp^\varrho}dQ=\int_\Theta\log\frac{dQ}{dP}dQ.$$

Combining the above equations yields (4). Furthermore, suppose there exists a sequence of partition $\{\Omega_i\}_{i\geq1}$ such that $\limsup\Omega_i=\Omega$, we have

$$\limsup_{\varrho_i}\int_\Theta\log\frac{dq^{\varrho_i}}{dp^{\varrho_i}}dq^{\varrho_i}=\limsup_{\varrho_i}\int_\Theta\log\frac{dq^{\varrho_i}}{dp^{\varrho_i}}dQ=\int_\Theta\log\frac{dq^\varrho}{dp^\varrho}dQ=\int_\Theta\log\frac{dq^\varrho}{dp^\varrho}dq^\varrho.$$

Hence $\limsup_{\varrho_i}\mathrm{KL}(q^{\varrho_i}\|p^{\varrho_i})=\mathrm{KL}(q^\varrho\parallel p^\varrho)$, which will be used in Appendix C.4.

C.2 Proof for (6)

By $p(X,Z)=p(Z|X)p(X)$, we have

$$\int\log\frac{p(X,Z^\varrho)}{q(Z^\varrho)}q(dZ^\varrho)=\log p(X)+\int\log\frac{p(Z^\varrho|X)}{q(Z^\varrho)}q(dZ^\varrho).$$

Taking the limit inferior on both sides, we have

$$\liminf_\varrho\int\log\frac{p(X,Z^\varrho)}{q(Z^\varrho)}q(dZ^\varrho)=\log p(X)-\limsup_\varrho\Big\{-\int\log\frac{p(Z^\varrho|X)}{q(Z^\varrho)}q(dZ^\varrho)\Big\}.$$

Combing the above equation with the definition of NPELBO in (5) and the KL divergence in (4) yields (6).

C.3 Derivation for (9)

By $p(G_0^\varrho, \{\boldsymbol{z}_j\}_{j=1}^J) = \int \cdots \int p\big(G_0^\varrho, \{G_j\}_{j=1}^J, \{\boldsymbol{z}_j\}_{j=1}^J\big) dG_1 dG_2 \cdots dG_J$ and the hierarchical generative structure, the evidence lower bound under partition Ω with respect to $q(G_0^\varrho)$ equals,

$$\begin{aligned}
& \mathrm{ELBO}^\varrho \\
= \; & \mathrm{E}_{q(G_0^\varrho)} \mathrm{E}_{q(\{\boldsymbol{z}_j\}_{j=1}^J)} \log p(G_0^\varrho, \{\boldsymbol{z}_j\}_{j=1}^J) - \mathrm{E}_{q(G_0^\varrho)} \log q(G_0^\varrho) + \text{constant} \\
= \; & \mathrm{E}_{q(G_0^\varrho)} \mathrm{E}_{q(\{\boldsymbol{z}_j\}_{j=1}^J)} \log q(G_0^\varrho) \prod_{j=1}^J \int p(G_j^\varrho | G_0^\varrho) p(\boldsymbol{z}_j | G_j^\varrho) dG_j - E_{q(G_0^\varrho)} \log q(G_0^\varrho) + \text{constant} \\
= \; & \sum_{j=1}^J \mathrm{E}_{q(G_0^\varrho)} \mathrm{E}_{q(\boldsymbol{z}_j)} \log \mathrm{E}_{p(G_j^\varrho | G_0^\varrho)} p(\boldsymbol{z}_j | G_j^\varrho) + \mathrm{E}_{q(G_0^\varrho)} \log p(G_0^\varrho) - \mathrm{E}_{q(G_0^\varrho)} \log q(G_0^\varrho) + \text{constant}.
\end{aligned}$$

Furthermore, based on the equation above, (8) can be expressed as $\mathrm{NPELBO} = \liminf_\varrho \mathrm{ELBO}^\varrho$.

C.4 Derivation for (12)

By the formula of moments for Dirichlet-distributed random variables, we obtain

$$\mathrm{E}_{p(G_s^\varrho | G_0^\varrho)} p(\hat{\boldsymbol{z}}_{s,t} | G_s^\varrho) = \frac{\Gamma(\gamma)}{\Gamma(\gamma + N_s)} \prod_{k=1}^K \frac{\Gamma(\gamma G_{0k} + \hat{n}_{sk,t})}{\Gamma(\gamma G_{0k})}.$$

Based on the points $\{\phi_k\}_{k=1}^K$ defined in Section 5.1, we propose a sequence of partition $\{\Omega_c : \Omega_c = \bigcup_{k=0}^K \Omega_{ck}\}_{c \geq 1}$ to approach Ω, where $\Omega_{ck} = (\phi_k - c^{-1}, \phi_k + c^{-1}]$ for $k = 1, \ldots, K$ and Ω_{c0} is the corresponding complement. Under Ω_c, $q(G_0^{\varrho_c}) = \mathrm{d}_{K+1}\big(m_0^{-1}(G_0^{\varrho_c} - M^{\varrho_c})\big)$ and $p(G_0^{\varrho_c}) = \mathrm{d}_{K+1}(G_0^{\varrho_c})$, where $\mathrm{d}_{K+1}(\cdot)$ denotes the density function for $(K+1)$-dimensional Dirichlet distribution, $M = \sum_{k=1}^K m_k \delta_{\phi_k}$ and M^{ϱ_c} is the corresponding induced random variable. By (10), the empirical evidence lower bound under Ω_c is

$$\begin{aligned}
& E_{q(G_0^{\varrho_c})} \Big\{ \sum_{k=1}^K (\alpha H_k^{\varrho_c} - 1) \log \frac{m_0 G_{0k}}{(G_{0k} - m_k)} + (\alpha H_0^{\varrho_c} - 1) \log m_0 \\
& + \sum_{s=1}^S \sum_{k=1}^K \sum_{t=1}^{T_s} \frac{J}{S T_s} \log \frac{\Gamma(\gamma G_{0k} + \hat{n}_{sk,t})}{\Gamma(\gamma G_{0k})} \Big\} + \text{constant},
\end{aligned}$$

where $H_k^{\varrho_c} = H(\Omega_{ck})$. Since $(G_{0k} - m_k)/m_0 \sim \mathrm{Beta}(H_k^{\varrho_c})$ under $q(G_0^{\varrho_c})$, the term $E_{q(G_0^{\varrho_c})}(\alpha H_k^{\varrho_c} - 1) \log m_0 (G_{0k} - m_k)^{-1}$ is constant with respect to parameters $\{m_k\}_{k=0}^K$. Taking lim sup on both sides of the above equation with $\limsup_{\varrho_c} E_{q(G_0^{\varrho_c})}(\log G_{0k}) = \log m_k$, $\limsup_{\varrho_c} H_k^{\varrho_c} = 0$ for $k > 0$ and $\limsup_{\varrho_c} H_0^{\varrho_c} = 1$, we obtain equation (12).

C.5 Justification for Section 4.3

In this section, we show that the empirical evidence lower bound achieves the limit inferior in NPELBO. With the partition $\Omega = (\phi_0, \phi_1, \ldots, \phi_K)$ defined in Section 4.3, there is no sampling within ϕ_0, and hence we have

$$q\big(G_0(\phi_0), G_0(\phi_1), \cdots, G_0(\phi_K)\big) \propto p\big(G_0(\phi_0), G_0(\phi_1), \cdots, G_0(\phi_K)\big) p\big(\boldsymbol{x} \mid G_0(\phi_1), \cdots, G_0(\phi_K)\big).$$

As the likelihood part $p\big(\boldsymbol{x} | G_0(\phi_1), \cdots, G_0(\phi_K)\big)$ does not contain $G(\phi_0)$, by integrating both sides with respect to $G_0(\phi_1), \cdots, G_0(\phi_K)$, we can get $q\big(G_0(\phi_0)\big) \propto p\big(G_0(\phi_0)\big)$. Moreover, the KL divergence between the variational distribution and true posterior is

$$\begin{aligned}
& \mathcal{KL}\big(q(G_0^\varrho) \parallel p(G_0^\varrho \mid \boldsymbol{x})\big) \\
= \; & \int \log \frac{q\big(G_0(\phi_0), G_0(\phi_1), \cdots, G_0(\phi_K)\big)}{p\big(G_0(\phi_0), G_0(\phi_1), \cdots, G_0(\phi_K)\big) p\big(\boldsymbol{x} \mid G_0(\phi_1), \cdots, G_0(\phi_K)\big)} dq\big(G_0(\phi_0), G_0(\phi_1), \cdots, G_0(\phi_K)\big) \\
= \; & -\log \mathcal{N},
\end{aligned}$$

because

$$q\big(G_0(\phi_0), G_0(\phi_1), \cdots, G_0(\phi_K)\big) = p\big(G_0(\phi_0), G_0(\phi_1), \cdots, G_0(\phi_K)\big) p\big(\boldsymbol{x} \mid G_0(\phi_1), \cdots, G_0(\phi_K)\big)/N,$$

where N is the normalization constant,

$$\begin{aligned}
\mathcal{N} &= \int \cdots \int p\big(G_0(\phi_0), G_0(\phi_1), \cdots, G_0(\phi_K)\big) p\big(\boldsymbol{x} \mid G_0(\phi_1), \cdots, G_0(\phi_K)\big) dG_0(\phi_1) dG_0(\phi_1) \cdots dG_0(\phi_K) \\
&= \int p\big(\boldsymbol{x} \mid G_0(\phi_1), \cdots, G_0(\phi_K)\big) dp\big(G_0(\phi_0), G_0(\phi_1), \cdots, G_0(\phi_K)\big) \\
&= \int p\big(\boldsymbol{x} \mid G_0(\phi_1), \cdots, G_0(\phi_K)\big) dp\big(G_0(\phi_1), \cdots, G_0(\phi_K)\big).
\end{aligned}$$

It is obvious that $\mathcal{N}$ is independent of $p\big(G_0(\phi_0)\big)$. Therefore, if we partition ϕ_0 into $\phi_0^1 \cup \phi_0^2$, the normalization constant $\mathcal{N}$ will not change, that is, the KL divergence under Ω and $\Omega' = (\phi_0^1, \phi_0^2, \phi_1, \ldots, \phi_K)$ are the same. Consequently, the partition Ω enables the limit superior of KL divergence to be reached. By (6), the limit inferior of NPELBO is also attained.

C.6 Derivation for (B.1)

Consider the Lagrange multiplier of constrained optimization,

$$L' = -\sum_{k=1}^{K} \log m_k + (\alpha - 1) \log m_0 + \sum_{s=1}^{S} \sum_{k=1}^{K} \frac{J}{ST_s} \sum_{t=1}^{T_s} \log \frac{\Gamma(\gamma m_k + \hat{n}_{sk,t})}{\Gamma(\gamma m_k)} - \lambda(\sum_{k=0}^{K} m_k - 1),$$

its first order conditions satisfy,

$$\begin{cases} JS^{-1}\gamma \sum_{s=1}^{S} \big\{T_s^{-1} \sum_{t=1}^{T_s} \Phi(\gamma m_k + \hat{n}_{sk,t}) - \Phi(\gamma m_k)\big\} m_k - 1 = m_k \lambda, & k = 1, \ldots, K, \\ \alpha - 1 = m_0 \lambda, & k = 0. \end{cases}$$

Dividing λ on both sides of the above equations, the definition of $\{m_k^*\}_{k=0}^K$ in (B.1) follows.

We next show that this updating is consistent with the gradient descent after the inverse logit transformation, that is, transforming $\{m_k\}_{k=0}^K$ by $m_k = e^{\theta_k} / \sum_{l=0}^K e^{\theta_l}$ to remove the constraint of $\sum_{k=0}^K m_k = 1$. By $\partial m_k / \partial \theta_k = m_k - m_k^2$, $\partial m_l / \partial \theta_k = -m_k m_l$ for $l \neq k$, and the chain rule, we have

$$\frac{\partial L}{\partial \theta_k} = \begin{cases} JS^{-1}\gamma \sum_{s=1}^{S} \big\{T_s^{-1} \sum_{t=1}^{T_s} \Phi(\gamma m_k + \hat{n}_{sk,t}) - \Phi(\gamma m_k)\big\} m_k - 1 - \Lambda m_k & k = 1, \ldots, K, \\ \alpha - 1 - \Lambda m_k & k = 0, \end{cases}$$

where L denotes $\widehat{\mathrm{ELBO}}^{\alpha}$ in (12) and

$$\Lambda = \alpha - 1 + \sum_{k=1}^{K} \Big[JS^{-1}\gamma \sum_{s=1}^{S} \big\{T_s^{-1} \sum_{t=1}^{T_s} \Phi(\gamma m_k + \hat{n}_{sk,t}) - \Phi(\gamma m_k)\big\} m_k - 1 \Big].$$

As $\partial L / \partial \theta_k = \Lambda(m_k^* - m_k)$, $(m_k^* - m_k)$ represents the gradient with respect to θ_k after the inverse logit transformation.

C.7 Derivation for (B.4)

For the HBNP model, we use an unnormalized random measure as the prior of G_0. Given moments for Dirichlet-distributed random variables, we obtain

$$\log \mathrm{E}_{p(G_s^\alpha | G_0^\alpha)} p(\hat{\boldsymbol{z}}_{s,t} | G_s^\alpha) = \log \frac{\Gamma(\sum_{k=0}^K G_{0k})}{\Gamma(\sum_{k=0}^K G_{0k} + N_s)} \prod_{k=1}^{K} \frac{\Gamma(G_{0k} + \hat{n}_{sk,t})}{\Gamma(G_{0k})},$$

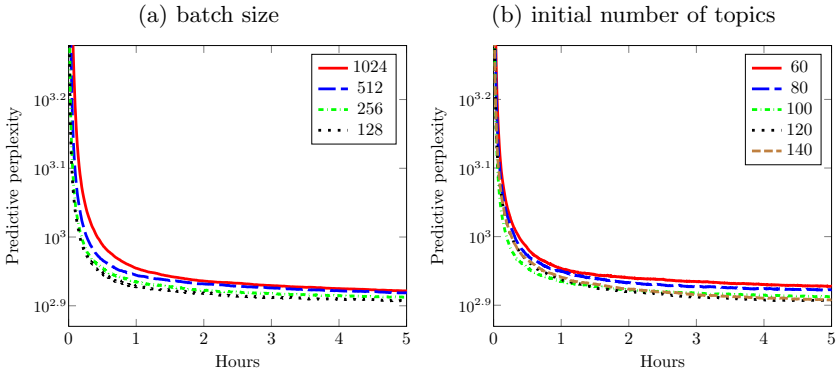


Figure 3: Plots for the perplexity vs running time for different batch sizes and initial numbers of topics.

In analogy to Appendix C.4, the empirical evidence lower bound under Ω_c is

$$K \log \mu + E_{q(G_0^{\Omega_c})} \Big\{ \sum_{k=1}^{K} \log p(G_{0k}^{\Omega_c}) + \log p(G_{00}^{\Omega_c}) + \frac{J}{S} \sum_{s=1}^{S} \sum_{k=1}^{K} T_s^{-1} \sum_{t=1}^{T_s} \log \frac{\Gamma(\gamma G_{0k}^{\Omega_c} + \hat{n}_{sk,t})}{\Gamma(\gamma G_{0k}^{\Omega_c})} \Big\},$$

up to a constant, where $K \log \mu$ comes from the Jacob matrix from $G_0, G_1, \ldots, G_K$ to $\mu, m_1, \ldots, m_K$. As the partition converges to single points and the corresponding complement, $\limsup_{\Omega_c} p(G_{0k}^{\Omega_c}) = v(G_{0k}^{\Omega_c})$ and $\limsup_{\Omega_c} p(G_{00}^{\Omega_c}) = u(G_{00}^{\Omega_c})$. Therefore, we can obtain (B.4) by $\limsup_{\Omega_c} G_{0k} = \mu m_k$ for $k \neq 0$ and $\limsup_{\Omega_c} G_{00} = \mu m_0$. Specially, for the ΓDP model, $\widehat{\text{ELBO}}^{\Omega}$ takes the form of

$$\mu - \sum_{k=1}^{K} \log m_k + (\alpha - 1) \log \mu m_0 + \frac{J}{S} \sum_{s=1}^{S} \Big\{ \log \frac{\Gamma(\mu)}{\Gamma(\mu + N_s)} + \sum_{k=1}^{K} \frac{1}{T_s} \sum_{t=1}^{T_s} \log \frac{\Gamma(\mu m_k + \hat{n}_{sk,t})}{\Gamma(\mu m_k)} \Big\},$$

up to a constant and (B.7) is also attained.

D Computational Complexity Analysis, Data and Code

For CATVI, updating the global variables takes linear time, and the Monte Carlo step iteratively samples each z_{ji} from K possible topics. Therefore, the computational complexity of Algorithm 1 is dominated by $O(K + \overline{T_s} K \overline{N_S})$, where $\overline{N_S}$ is the average number of words in a document, and $\overline{T_s}$ is the average of T_s defined in Algorithm 1. To implement this algorithm, we conduct our experiments on a c5d.4xlarge instance on the AWS EC2 platform, with 16 vCPUs and 32 GB RAM. It takes at most 5 hours to run all numerical experiments.

Python code for CATVI is available at `https://github.com/yiruiliu110/ConditionalVI`.We obtain the *arXiv* and *Wiki* data from public open resources `https://arxiv.org/help/bulk_data` and `https://dumps.wikimedia.org`, respectively. The *NYT* data are from Sandhaus (2008). For the comparison methods, we implement OVI using the Python package 'gensim.models.hdpmodel' under GNU Lesser general public license v2.1. Moreover, we implement MOVI and SMVI using the Python package 'bnpy' under 3-clause BSD license, which is available at `https://github.com/bnpy/bnpy`. Finally, the codes to run GS are available at https://github.com/linkstrife/HDP.

E Sensitivity Analysis Results of CATVI

Figure 3a plots the sensitivity analysis with respect to the batch size varying from 128 to 1024. Figure 3b plots the sensitivity analysis with respect to the initial number of topics varying from 60 to 140. We observe that the performance is not sensitive to the change of these hyperparameters.

Chapter 3

Bayesian Nonparametric for Graph Data

This chapter is dedicated to an article published at the 26th International Conference on Artificial Intelligence and Statistics, available online at `https://proceedings.mlr.press/v206/liu23a`.

EEGNN: Edge Enhanced Graph Neural Network with a Bayesian Nonparametric Graph Model

Yirui Liu[1,2] **Xinghao Qiao**[1] **Liying Wang**[3] **Jessica Lam**[4]
[1]London School of Economics and Political Science [2]J.P. Morgan
[3]Bayes Business School, City, University of London [4]University of Oxford

Abstract

Training deep graph neural networks (GNNs) poses a challenging task, as the performance of GNNs may suffer from the number of hidden message-passing layers. The literature has focused on the proposals of over-smoothing and under-reaching to explain the performance deterioration of deep GNNs. In this paper, we propose a new explanation for such deteriorated performance phenomenon, mis-simplification, that is, mistakenly simplifying graphs by preventing self-loops and forcing edges to be unweighted. We show that such simplifying can reduce the potential of message-passing layers to capture the structural information of graphs. In view of this, we propose a new framework, edge enhanced graph neural network (EEGNN). EEGNN uses the structural information extracted from the proposed Dirichlet mixture Poisson graph model (DMPGM), a Bayesian nonparametric model for graphs, to improve the performance of various deep message-passing GNNs. We propose a Markov chain Monte Carlo inference framework for DMPGM. Experiments over different datasets show that our method achieves considerable performance increase compared to baselines.

1 INTRODUCTION

Graph neural networks (GNNs) (Zhou et al., 2020; Wu et al., 2020) are important tools for analyzing graph data, such as social network (You et al., 2020), transportation network (Chen et al., 2021a), molecular graph (Huang et al., 2020), biological network (Zhang et al., 2021), financial transaction network (Wang et al., 2021), academic citation graph (Xu et al., 2021), and knowledge graph (Ji et al., 2021). GNNs have become popular with their state-of-the-art performance by applying deep learning methodologies to graphs. Among them, message passing neural networks (MPNN) (Gilmer et al., 2017) uses message-passing layers to compute node embeddings. Examples of MPNNs include graph convolutional neural networks (GCN) (Kipf and Welling, 2017), GraphSAGE (Hamilton et al., 2017), graph attention networks (GAT) (Veličković et al., 2018), and gated graph neural networks (GGNN) (Li et al., 2016). Similar to standard multi-layer perceptron (MLP) in deep learning, the message passing layer in a GNN framework aggregates information from the local neighbors of each node, and then transforms the information via an activation function into the embedding (Hamilton, 2020). A node embedding can aggregate information over N hop neighbors, in the form of N hidden message-passing layers, thus incorporating further reaches of the graph.

Although deeper layers in non-graph neural networks often achieve better performance (Krizhevsky et al., 2012; He et al., 2016), GNNs typically perform best with only 2 to 4 hop neighbors, that is, 2 to 4 hidden layers. In contrast, using a larger number of layers, termed as deep stacking, may lead to a substantial drop in the performance for GNNs (Klicpera et al., 2019; Rong et al., 2019; Li et al., 2020; Chen et al., 2020b). One explanation for this phenomenon is the *over-smoothing*. By applying graph convolution repeatedly over many hidden layers, the representation of the nodes will be indistinguishable. As a result, the *over-smoothing* can jeopardize the performance of deep GNNs. Another explanation is the *under-reaching*. When GNNs aggregate messages over long paths, the information propagation across distant nodes in the graph becomes difficult because it is susceptible to bottlenecks (Alon and Yahav, 2020). This causes GNNs to perform poorly in predicting tasks that require remote interaction (Singh et al., 2021; Hwang et al., 2021).

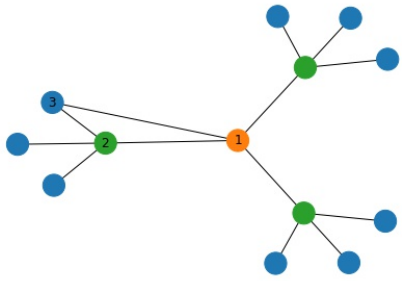
Many efforts have been devoted to addressing these limitations. To handle the over-smoothing, DropEdge (Rong et al., 2019) and DropNode (Huang et al., 2021) were proposed to randomly remove a certain number of edges or nodes from

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

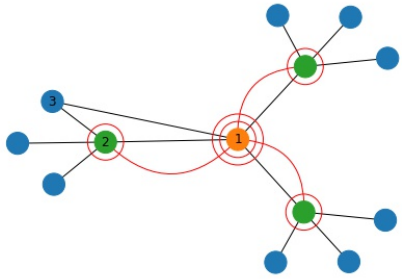
the input graph at each training epoch. These methods are likened to Dropout (Srivastava et al., 2014), which randomly drops hidden neurons in neural networks to prevent overfitting. On the contrary, to address the under-reaching, virtual edges (Gilmer et al., 2017), super nodes (Scarselli et al., 2009; Hwang et al., 2021), or short-cut edges (Allamanis et al., 2018) can be added to the original graph. However, none of the aforementioned methods consider adding or removing based on the structural information of the graph. Instead, the pattern of deciding which nodes or edges to be added or removed comes from an arbitrarily random selection. Although dropout has been effective in non-graph neural networks, its random removal and addition of nodes can disturb the graph structure, thus compromising the performance of GNNs that relies on the structure to propagate information.

Different from the *over-smoothing* and *under-reaching*, we propose a new explanation for performance deterioration of deeper GNNs from the perspective of misusing edge structural information, *mis-simplification*, explained as follows. Most observed graphs are recorded as simple graphs, where self-loops are not allowed, and all edges are unweighted and undirected (Shafie, 2015). In a natural way, GNNs are designed for learning such simple graphs that can be constructed by collapsing multiple edges into a single edge as well as removing self-loops. This approach, however, discards the information inherent in the original network. Take one example, for a source node connected to many neighboring target nodes (see node 1 in Figure 1a), its self-loop has an equal weight to neighboring non-loop edge, which may under-weigh the importance of this node. Take another example, no matter how similar the two nodes are (see nodes 1 and 2 in Figure 1a), only one edge is allowed to connect the pair of nodes, and as a result, the information passing between both nodes is restricted. Furthermore, edge $(1, 2)$ should play a more important role in message passing than edge $(1, 3)$, because node 2 is a key node with 3 sub-nodes in total, while node 3 is just a sub-node of node 2. However, typical GNNs treat these two edges indifferently as they are equally weighted in the simple graph. Therefore, such *mis-simplification* can reduce the potential of message-passing layers to capture structural information in GNNs.

To solve this issue, we propose an edge-enhanced graph neural network (EEGNN), which incorporates edge structural information in the message-passing layer. First, we assume that there is an underlying *virtual multigraph*, allowing for self-loops and for multiple edges between pairs of nodes, and the observed graph model can be viewed as a transformation of the virtual multigraph. As illustrated in Figure 1, the above Figure 1ais the original observed simple graph, while the below Figure 1b is the corresponding virtual graph. Second, to build the virtual multigraph that can capture the edge structural information, we propose the Dirichlet mixture Poisson graph model, a Bayesian non-



(a) The observed simple graph



(b) The corresponding virtual multigraph

Figure 1: The observed simple graph versus the virtual multigraph. The red edges are virtual edges in the virtual multigraph. In particular, the red circles are virtual self-loops.

parametric model. Following Caron and Fox (2017), the interactions between nodes are modelled by assigning a *sociability* parameter to each node. Then, the counts of edges are generated from a Poisson distribution, where the Poisson rate is the product of sociability parameters of the nodes in two ends. Finally, in the framework of EEGNN, we can then replace the observed graph in a GNN with the virtual multigraph. In this architecture, message-passing layers can then assign weights proportionally to the importance of the edges, thus passing the information from nodes to nodes in a more reasonable manner.

The main contribution of our paper is fourfold.

- We outline a new explanation for the poor performance of deep GNNs;
- We propose a new way to enhance existing GNN methods by utilizing the structural information of graphs;
- We propose a Bayesian nonparametric graph model and its Monte Carlo Markov chain (MCMC) inference procedure;
- We demonstrate the superior sample performance of our proposal over existing methods through the experiments on six real datasets and a financial application.

2 Preliminaries

2.1 GNN and Message Passing Layer

We begin by introducing some notation. Let $G = (V, E)$ be a graph with node set $V = \{v_1, \ldots, v_{|V|}\}$ and edge set $E = \{e_1, \ldots, e_{|E|}\}$, where $|V|$ and $|E|$ denote the number of nodes and edges in G, respectively. The adjacency matrix $A \in \mathbb{R}^{|V|\times|V|}$ is defined as $A_{ij} = 1$ if $(v_i, v_j) \in E$ and 0 otherwise. The corresponding degree matrix $D \in \mathbb{R}^{|V|\times|V|}$ is defined as $D = \text{diag}(D_1, \ldots, D_{|V|})$, where $D_i = \sum_{j=1}^{|V|} A_{ij}$. We denote the data matrix by $X \in R^{m\times|V|}$, whose j-th column corresponds to a m-dimensional feature vector of node j.

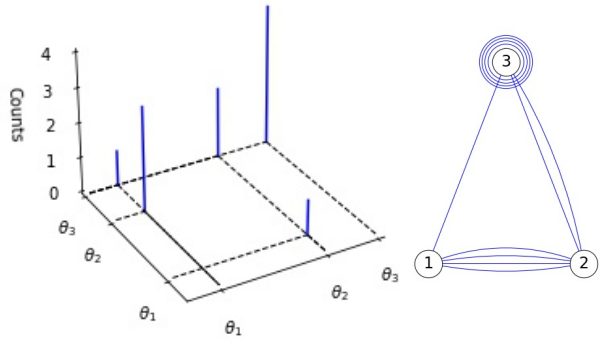
GNN is a neural network model to process graphs for node classification, edge prediction and graph classification (Gori et al., 2005; Zhou et al., 2020; Wang et al., 2021). Within various GNNs, information is exchanged between nodes and is updated by neural networks via message passing layers (Gilmer et al., 2017). Specifically, the initial representation, h_i^0 for node i, is generated by a function of this node's features. Then, the message passing layers update the representation based on this node's neighbors. The message passing contains two steps: the aggregation step and the update step. Denote the representation for node i in layer l by h_i^l. A message passing layer in GNN can be expressed as

$$h_i^{l+1} = \text{UPDATE}(h_i^l, \text{AGGREGATE}(h_j^l \mid j \in \mathcal{N}_i)), \quad (1)$$

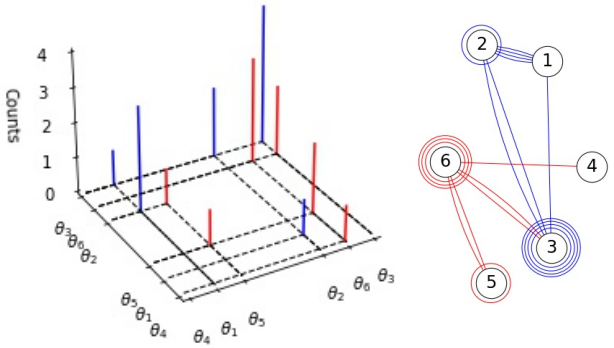
where AGGREGATE$(\cdot)$ denotes a permutation-invariant function, such as the sum, mean, and maximum, to send information from one node to another through edges, and UPDATE$(\cdot)$ denotes linear or nonlinear differentiable functions such as MLP, $\mathcal{N}_i$ denotes the neighborhood of node i, that is, the set of nodes directly connected to node i. For example, the vanilla GCN uses $h_i^{l+1} = \sigma(\sum_{j=1}^{|V|} \tilde{P}_{ij} h_j^{l+1} W^l \mid j \in \mathcal{N}_i \cup \{i\})$, or in matrix form, $H^{l+1} = \sigma(\tilde{P} H^l W^l)$ (Kipf and Welling, 2017), APPNP uses $H^{l+1} = (1-\alpha)\tilde{P}H^l + \alpha H^0$ (Klicpera et al., 2019), and GCNII uses $H^{l+1} = \sigma\big((1-\alpha)\tilde{P}H^l + \alpha H^0)((1-\beta)I_n + \beta W^l)\big)$ (Chen et al., 2020b), where $\tilde{P} = (D+I)^{-\frac{1}{2}}(A+I)(D+I)^{-\frac{1}{2}}$, H^l is the representation for all nodes in layer l, σ and W^l are respectively the activation function and the corresponding weight in a neural network layer, and α and β are hyperparameters. The formulas above show that the GNN treats each edge with equal weight and hence leads to *mis-simplification*. In order to solve this issue, we adopt a Bayesian nonparametric sparse graph model to generate the virtual edges and virtual multigraph.

2.2 Bayesian Nonparametric Sparse Graph Model

In contrast with other graph models that are based on node feature embeddings, Caron and Fox (2017) represent the observed graph G as a point process on $\mathbb{R}^2$,



(a) Graph model in Caron and Fox (2017)



(b) Dirichlet mixture Poisson graph model (DMPGM)

Figure 2: Bayesian nonparametric graph model. Figure 2a illustrates the model in Caron and Fox (2017), while Figure 2b illustrates the proposed graph model in this paper. The left sub-figures show the proxy for nodes, θ_is, and the number of edges among them. The right sub-figures display the corresponding multigraph. In Figure 2b, red and blue are used to indicate two clusters in the edges. The circles around nodes denote self-loops. Finally, the number of circles or links denotes the multiplicity.

$G = \sum_{i,j} z_{i,j} \delta_{(\theta_i,\theta_j)}$, where Dirac function $\delta_{(\theta_i,\theta_j)}$ is equal to 1 at (θ_i, θ_j) and equal to 0 elsewhere, $z_{i,j}$ is the multiplicity for edge (i, j), and θ_i is a proxy for node i on the real axis, as illustrated in Figure 2a. Note that the same definition for node i is also applied to node j, but we omit the explanation for node j to avoid redundancy. This representation specifies the source and target nodes for each edge. To model the possibility for two nodes constructing an edge, a sociability parameter $w_i > 0$ is assigned to node i for each $i = 1, \ldots, |V|$. Following Aldous (1997), the graph model can be factorized as $p(A_{ij} = 1) = 1 - \exp(-2w_i w_j)$ for $i \neq j$ and $p(A_{ii} = 1) = 1 - \exp(-w_i^2)$ otherwise. This is equivalent to modelling an unobserved integer-valued multigraph as $z_{ij} \sim \text{Poisson}(w_i w_j)$ and setting $A_{ij} = \mathbb{1}_{z_{ij}+z_{ji}>0}$. To model the sparsity property in real graphs, that is, $|E| = o\big(|V|^2\big)$, the sociability is generated from a completely random measure with infinite activity

(Caron and Fox, 2017), such as gamma process, stable process and inverse Gaussian process (Ghosal and Van der Vaart, 2017). See also Appendix A for a short review of completely random measures. This model allows for self-loop and multi-edges, and thus can be used to build a virtual multigraph. However, as the Poisson intensity is factorized as the outer product of a vector with itself, only one feature for each node is considered, which restricts the capability of this model in confronting real data. To address this issue, we propose a novel model in Section 3.1 below.

3 METHODOLOGY

3.1 Dirichlet Mixture Poisson Graph Model

To adopt the latent community information among nodes in the graph, mixed-membership stochastic block model (Airoldi et al., 2008) associates each node with latent cluster distributions. In an analogy, we add cluster-membership features to each pair of edges instead of nodes. Specifically, we extend the graph model in Caron and Fox (2017) by proposing the following Dirichlet mixture Poisson graph model (DMPGM),

$$\begin{aligned}
\pi = (\pi_1, \pi_2, \cdots) &\sim \text{GEM}(\alpha), \\
W_0 = \sum_{i=1}^{\infty} w_{0,i}\delta_{\theta_i} &\sim \text{CRM}(\kappa, v), \\
W_k = \sum_{i=1}^{\infty} w_{k,i}\delta_{\theta_i} &\sim \Gamma\text{P}(W_0), \\
z_{ij} \sim \text{Poisson}(\sum_{k=1}^{\infty} \pi_k w_{k,i} \times w_{k,j}&), \\
A_{ij} = \min(z_{ij} + z_{ji}, 1)\mathbb{1}_{i\neq j}&,
\end{aligned} \tag{2}$$

for $i, j, k \in \mathbb{N}^+$, where GEM(α) is the distribution for atom sizes of a Dirichlet process DP(α) and each atom corresponds to a distinct cluster. Moreover, CRM(κ, v) denotes a completely random measure with $v^c(dw, d\theta) = \kappa(d\theta)v(dw)$ as its Levy measure, and $\Gamma\text{P}(H)$ denotes gamma process with the base measure H. See Appendix A for details of these stochastic processes. We summarize the probabilistic generative steps as follows. First, the cluster distribution π is assigned with a prior GEM(α), which allows for infinitely many clusters. Second, we use a hierarchical structure to generate values for the node sociability parameter in each cluster. W_0, sampled from a completely random measure, is used as the base measure in $\Gamma\text{P}(W_0)$ for W_k such that $w_{k,i}$ belongs to gamma distribution parameterized by $w_{0,i}$, $w_{k,i} \sim \text{Gamma}(w_{0,i})$. This hierarchical setting is designed to ensure the components in W_k share atom locations (Teh et al., 2006; Liu et al., 2022). Finally, following Caron and Fox (2017), an undirected multigraph $\sum_{i,j} z_{ij}\delta(\theta_i, \theta_j)$ is generated from a Poisson process, where z_{ij} is the Poisson-distributed multiplicity for edge (i, j). By aggregating multiple edges to a single edge for each pair of nodes and removing self-loops, a simple graph is transformed from the multigraph. The corresponding adjacent matrix $A = (A_{ij})$ to the observable simple graph can then be generated. An example of DMPGM is illustrated in Figure 2b.

DMPGM can be equivalently expressed under a mixture model framework. Specifically, a set of edges in each cluster is sampled from Poisson($\pi_k\bar{w}_k^2$), where $\bar{w}_k = \sum_{i=1}^{\infty} w_{k,i}$. As a consequence, this is equivalent to sampling the total number of edges n from Poisson(λ) with $\lambda = \sum_{k=1}^{\infty} \pi_k\bar{w}_k^2$, and then assigning each edge a cluster membership from Categorical$\big(\frac{\pi_1\bar{w}_1^2}{\lambda}, \frac{\pi_2\bar{w}_2^2}{\lambda}, \dots\big)$. Following the same methodology, for each edge, a pair of nodes is then sampled from Categorical$(\frac{w_{k,1}}{\bar{w}_k}, \frac{w_{k,2}}{\bar{w}_k}, \dots)$ in the cluster k. Hence, a relationship between edges and nodes is constructed. We summarise this equivalent expression for DMPGM as follows,

$$\begin{aligned}
n &\sim \text{Poisson}(\sum_{k=1}^{\infty} \pi_k\bar{w}_k^2), \\
k &\sim \text{Categorical}\big(\frac{\pi_1\bar{w}_1^2}{\lambda}, \frac{\pi_2\bar{w}_2^2}{\lambda}, \dots\big), \\
i, j &\sim \text{Categorical}\big(\frac{w_{k,1}}{\bar{w}_k}, \frac{w_{k,2}}{\bar{w}_k}, \dots\big),
\end{aligned} \tag{3}$$

where other structures in equation (2) remain the same. In Appendix C, we show that DMPGM enjoys similar properties as the model in Caron and Fox (2017) in the following theorem.

Theorem 1 *The graph constructed by DMPGM is sparse if CRM in (2) has infinite activity.*

For example, using the gamma process as the completely random measure leads to a sparse graph in the DMPGM, which makes it more effective for modeling real-life data.

It is worth noting that DMPGM extends the model in Caron and Fox (2017) by assuming that edges can belong to different clusters. As a result, DMPGM is more flexible and applicable in modelling real data. We also note that DMPGM is distinct from the overlapping communities graph model (Todeschini et al., 2020) and graph Poisson factorization (Zhou, 2015), because we assign a Dirichlet prior for the clustering distribution, and hence can allow a nonparametric estimation of the number of edge clusters. Moreover, Williamson (2016) uses the hierarchical Dirichlet process (HDP) (Teh et al., 2006) to construct the graph. However, as HDP only models the node distribution within a cluster, the number of edges is ignored. As a result, this model cannot be used for EEGNN framework. Finally, a generative model that shares some fundamental similarities with DMPGM is proposed by Ricci et al. (2022). However, this concurrent work does not investigate the use of a Bayesian nonparametric graph model to improve GNNs.

3.2 MCMC Inference Framework

We propose a detailed MCMC framework to infer the posteriors for DMPGM in a nonparametric way. Following Caron and Fox (2017) and Liu et al. (2022), the posterior distribution for $W_k = \sum_{i=1}^{\infty} w_{k,i}\delta_{\theta_i}$, $k \geqslant 0$, are restricted to the weights $\{w_{k,i}\}$ because the locations $\{\theta_i\}$ of both observed and unobserved nodes are not likelihood identifiable, thus being ignored. Moreover, given the observed nodes set V, the weights for each W_k are truncated to a $(|V|+1)$-dimensional vector, $\boldsymbol{w}_k = (w_{k,0}, w_{k,1}, \ldots, w_{k,|V|})^{\mathrm{T}}$, where $w_{k,i}$ corresponds to the weight on an observed node i for $1 \leqslant i \leqslant |V|$, and $w_{k,0}$ is the sum of weights for all unobserved nodes. Similarly, the posterior distribution for π is truncated to a $(K+1)$-dimensional vector, $\boldsymbol{\pi} = (\pi_0, \pi_1, \ldots, \pi_K)^{\mathrm{T}}$, where K is the truncated number of clusters and is inferred adaptively in Step 4 below, and π_0 corresponds to the cluster without any observation. Consequently, given the truncation levels $|V|$ and K for W_0 and π, respectively, DMPGM contains the following parameters to infer: $\boldsymbol{\pi}$, $\{\boldsymbol{w}_k\}_{k\geqslant 0}$, $\boldsymbol{z} = \{z_{ij}\}_{A_{ij}=1}$ and cluster membership $\boldsymbol{c} = \{c_{ijl}\}_{A_{ij}=1, 1\leqslant l\leqslant z_{ij}}$.

We next propose a MCMC inference framework that can infer the number of edge clusters in a Bayesian nonparametric manner in the following steps.

Step 1 Update $w_{0,1}, \ldots, w_{0,|V|} | \bar{w}_0, \boldsymbol{z}, \boldsymbol{c}$ using Hamiltonian Monte Carlo (Kroese et al., 2011), where $\bar{w}_0 = \sum_{i=0}^{|V|} w_{0,i}$ with the log-posterior and its gradient provided in Appendix B.1.

Step 2 Update $w_{k,1}, \ldots, w_{k,|V|} | \bar{w}_k, w_0, \boldsymbol{z}, \boldsymbol{c}$ for $k = 1, \ldots, K$ given the conjugacy, where $\bar{w}_k = \sum_{i=0}^{|V|} w_{k,i}$. We sample $\widetilde{w}_{k,i} \sim \text{Dirichlet}(\nu_0, \nu_1, \ldots, \nu_{|V|})$, where $\nu_i = w_{0,i} + \sum_{j=1}^{|V|} n_{k,i}$, $n_{k,i} = \sum_{j=1}^{|V|}\sum_{l=1}^{z_{ij}} \{\mathbb{1}_{c_{ijl}=k} + \mathbb{1}_{c_{jil}=k}\}$, and then compute $w_{k,i} = \bar{w}_k \widetilde{w}_{k,i}$.

Step 3 Update $\pi_0, \pi_1, \ldots, \pi_K | \boldsymbol{z}, \boldsymbol{c}$ using the conjugacy. Analogous to Step 2, we sample $\pi_k \sim \text{Dirichlet}(n_0, n_1, \ldots, n_k)$, where $n_k = \sum_{i=1}^{|V|}\sum_{j=1}^{|V|}\sum_{l=1}^{z_{ij}} \mathbb{1}_{c_{ijl}=k}$ for $k > 0$ and $n_0 = \alpha$.

Step 4 Update the latent edge cluster membership $c_{ijl} | \{\boldsymbol{w}_k\}_{k\geqslant 0}, \boldsymbol{\pi}$ for each pair (i,j) such that $A_{ij} = 1$ and for $l = 1, \ldots, z_{ij}$. For each edge we sample from the multinomial distribution $p(c_{ijl} = k) \propto \pi_k w_{k,i} w_{k,j}$ for $k = 0, 1 \ldots, K$. In this step, if $k = 0$ is sampled, we add a new cluster (Teh et al., 2006; Bryant and Sudderth, 2012; Liu et al., 2022), and increase the truncated number of clusters from K to $K+1$.

Step 5 Update the unobserved $z_{ij} \mid \boldsymbol{\pi}, \boldsymbol{w}_k \sim$ Truncated–Poisson$(\sum_{k=0}^{K} \pi_k w_{k,i} w_{k,j})$ for each pair (i,j) such that $A_{ij} = 1$, where truncated Poisson is a conditional probability distribution of a Poisson-distributed random variable with strictly positive counts (Cohen, 1960).

Step 6 Update the $\bar{w}_k$ and $\bar{w}_0$ using Metropolis–Hastings (Kroese et al., 2011) algorithm based on the log-posterior provided in Appendix B.2.

We iterate over Steps 1–6 until convergence. For the MCMC algorithm, the global variables are updated in linear time, and the Monte Carlo step iteratively samples from K clusters. Therefore, the computational complexity is dominated by $O\big(K \max\{|V|, |E|\}\big)$.

3.3 Edge Enhanced Message Passing

In conventional message passing layers built from a simple graph, information for node i is obtained from edges connected to its neighboring nodes in $\mathcal{N}_i$ and from its self-loop. In these layers, each edge (i,j) for $j \in \mathcal{N}_i \cup \{i\}$ has equal weight, resulting in *mis-simplification* of the more complex structural information for the GNN, as described in Section 1. To overcome this *mis-simplification*, we sample artificial edges given the estimated DMPGM, from which we construct a virtual multigraph

$$G^* = (V, E, r), \;\; r\big((i,j)\big) = z_{ij}, \tag{4}$$

where the multiplicity-map $r : E \to \mathbb{N}^+$ assigns to each edge an integer to represent its multiplicity, and z_{ij} in DMPGM is defined in (2) and is inferred from Step 5 in Section 3.2. In this way, we can extract the edge structural information, via the inferred multiplicity for each edge, using the DMPGM model to build the virtual multigraph. For example, as illustrated in Figure 1, two artificial self-loops are added to nodes 1, one artificial self-loop is added to nodes 2, and the edge $(1,2)$ is assigned with multiplicity 2, where the multiplicity is determined by $z_{11} = 2$, $z_{22} = 1$ and $z_{12} + z_{21} = 2$, respectively. We then replace the original simple graph in the message passing layers by the generated virtual multigraph, that is,

$$\begin{aligned} h_i^{l+1} =& \text{UPDATE}^l(h_i^l, r(i,i), \\ & \text{AGGREGATE}^l(h_j^l, r(i,j) \mid j \in \mathcal{N}_i)). \end{aligned} \tag{5}$$

For example, for GCN, APPNP and GCNII, we replace $\tilde{P}$ by $\hat{P}$, where $\hat{P}$ is defined as

$$\begin{aligned} \hat{P} =& \hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}}, \;\; \hat{A} = (\hat{A}_{ij} = z_{ij}), \\ & \hat{D} = \text{diag}(\hat{D}_i = \sum_j z_{ij}). \end{aligned} \tag{6}$$

In addition, as the virtual multigraph already contains self-loops, there is no need to add the self-loops again to the message passing layers. This is different from conventional GNNs, where the self-loops are often added and forced to be a single edge. Though GIN (Xu et al., 2019) and

JKNet (Xu et al., 2018) also assign different weights for self-loops empirically, we are the first to propose a method to systematically determine the relative weights for self-loops and other edges.

In summary, conditional on the updated parameters of DMPGM in each iteration, we sample the multiplicity of each edge. Then a set of multiedges and self-loops are generated from DMPGM, which can be used to build a virtual graph and update GNN trainable parameters. We present the proposed EEGNN algorithm in Algorithm 1.

Algorithm 1: EEGNN Algorithm

Iterate Step 1 to Step 6 in Section 3.2 till the MCMC chains converge.
Set up initialization of trainable parameters in EEGNN.
repeat
1. Build the virtual graph and sample $\hat{P}$ according to (6),
2. Use $\hat{P}$ to replace $\tilde{P}$,
3. Update GNN parameters using the gradient descent,
4. Obtain a new sampling for the parameters in DMPGM by implementing Step 1 to Step 6,

until the convergence of the loss function of EEGNN

It is worth noting that we opted not to employ the stochastic block model in our study as it produces only dense graphs where the number of edges increases proportionally to the square of the number of nodes, whereas real-world networks tend to be sparse (Caron and Fox, 2017). Moreover, the stochastic block model does not allow for constructing a virtual multi-graph on edges. To build the virtual multi-graph, it is needed to use a statistical model on edges instead of on nodes.

3.4 Comparison with Other Methods

Our proposed method, EEGNN, addresses the performance deterioration of deep GNNs by using the structural information extracted from a Bayesian nonparametric graph model, DMPGM, to improve the performance of various deep message-passing GNNs. This is in contrast to relevant methods such as the attention and edge-label guided GNNs (Zhou et al., 2022) and edge-enhanced graph convolution networks (Cui et al., 2020), which focus on integrating syntactic dependency or dependency label information into GCN to perform event detection or named entity recognition, respectively. Edge-feature-enhanced GNNs (Gong and Cheng, 2019), another competing method, focuses on integrating edge features instead of extracting edge information based on the observed graph in an unsupervised-learning fashion. Our EEGNN framework differs from these methods as it addresses the issue of *mis-simplification* in deep GNNs and uses structural information from DMPGM to improve performance.

4 EXPERIMENTS

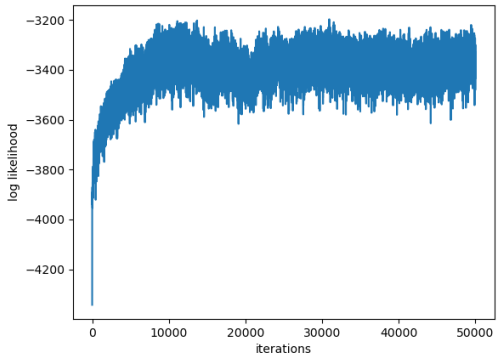
4.1 Datasets and Bayesian Estimation

In this section, we demonstrate through real data examples that EEGNN can effectively use the edge structural information to improve the performance for various GNNs. We conduct empirical experiments to compare EEGNN with representative baselines across six well-established network datasets. First, *Cora*, *Citeseer*, and *PubMed* are standard benchmark datasets for citation networks (Yang et al., 2016). In these networks, nodes represent papers, and edges indicate cross citations between papers. Node features are the bag-of-words embedding of the contents, and node labels are academic subjects. Second, *Texas*, *Cornell*, and *Wisconsin* are webpage cross-link networks (Pei et al., 2020). Their nodes represent web pages of universities, and edges represent hyperlinks between them. Node features are bag-of-words embedding of the websites. Node labels contain five categories for the webs including students, projects, courses, staff, and faculty. Statistics for these datasets are summarized in in Table 1.

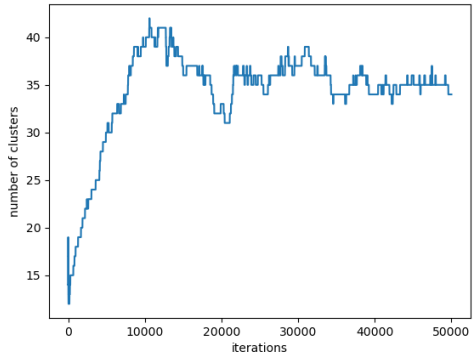
Table 1: Graph datasets statistics.

Dataset	*Cora*	*Citeseer*	*PubMed*	*Texas*	*Wisconsin*	*Cornell*
Nodes	2,708	3,327	19,717	183	183	183
Edges	5,429	4,732	44,338	309	499	295
Degrees	3.88	2.84	4.50	3.38	5.45	3.22
Features	1,433	3,703	500	1,703	1,703	1,703
Classes	7	6	3	5	5	5

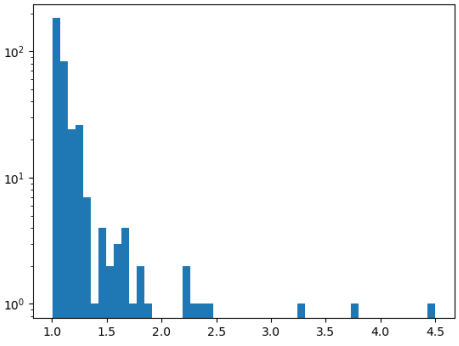
Our experiments are implemented by using a gamma process as the completely random measure in (2). Following Section 3.2, we infer the parameters of DMPGM using MCMC in the following way. We use population based training (Jaderberg et al., 2017) to tune the hyperparameters in DMPGM. For each dataset, we grow the MCMC chain up to 50,000 epochs. Figures 3a and 3b display the log-likelihood and number of clusters with respect to training epochs for the *Texas* dataset. (The training results for the other datasets are shown in Appendix D.) Figure 3a shows that the log-likelihood of *Texas* for the DMPGM converges after 10,000 epochs. Moreover, benefiting from the Bayesian nonparametric model, we can infer the number of edge clusters in a data-adaptive manner (Liu et al., 2022). Figure 3b shows that the inferred number of edges per node (termed as *multiplicity* of virtual edges per node) rises from an initial value of 10 to 50 at the start of training and then converges to around 35. The inferred edge multiplicity is displayed in the histograms in Figure 3c. These histograms show that a large proportion of the edges in the observed graph have underlying multi-edges, suggesting the *mis-simplification* in the original observed graph.



(a) The training log likelihood of DMPGM



(b) The inferred number of edge clusters



(c) Histogram of the expected multiplicity of virtual edges

Figure 3: The MCMC inference results for *Texas*.

4.2 Comparison with Baselines

With the inference results of DMPGM, following Section 3.3, we implement the experiments to compare baseline GNNs and their edge enhanced versions. For the baseline GNNs, we chose SGC (Wu et al., 2019) and its variant, including APPNP (Klicpera et al., 2019) and GCNII (Chen et al., 2020a), and hence name their edge enhanced versions as EE-SGC, EE-APPNP and EE-GCNII, respectively. To make a fair comparison, we follow the settings of the 'sweet point' GNN hyperparameter configuration in Chen et al. (2021b) for all datasets. The details of these hyperparameter settings are collected in Appendix E. For all experiments, the GNNS are trained with a maximum of 1000 epochs and an early stopping patience of 100 epochs. To analyze the effect of EEGNN with different numbers of layers, we run the experiments for 2, 16, 32 and 64 layers. We randomly split node features in each dataset into training and test sets, train the baseline GNNs and the edge enhanced versions using the same training set, and then compute the node clustering accuracy on the test set. In the transductive learning framework for GNNs, it is noted that the edge information is not partitioned as described in Kipf and Welling (2017). We repeat this procedure 50 times for each model and dataset. The mean predictive accuracy and the corresponding standard deviation are reported in Table 2.

We observe a few apparent patterns. First, EEGNN can improve the performance of the baseline models in most cases. For example, SGC, the backbone GNN for various models, performs poorly with 32 layers (see Table 2c). However, with the aid of our EEGNN framework, the accuracy of the SGC model is increased by more than 6% for *Cora*, and by approximately 2% across other candidate datasets. Moreover, SGC performs even worse with 64 layers for *Cora* and *Pubmed* (see Table 2d). EEGNNs largely improve the prediction accuracy in both cases, by 9.89% and 23.30%, respectively. It is worth noting that the improvements are attained without changing any other settings. As using virtual multigraph or observed simple graph brings in the only difference, this provides strong evidence to reveal that EEGNN can be used as a tool to enhance baseline GNNs by alleviating the *mis-simplification* problem.

Second, for APPNP and GCNII, EEGNNs achieve similar accuracies on the *Cora*, *Citeseer* and *PubMed* datasets, but substantially improve the performance on *Texas*, *Wisconsin* and *Cornell*. Especially, with 64 layers, EE-GCNII for *Texas* leads to more than 6% improvement, and EE-APPNP for *Citeseer* results in more than 10% increase in the predictive accuracy. On the other hand, as APPNP and GCNII have already reached relatively high accuracy (approximately $70\% - 80\%$) on the *Cora*, *Citeseer* and *PubMed*, further enhancement to higher accuracy tends to be difficult.

Finally, we observe that EEGNN has a larger impact on the performance of deeper SGC on the *Cora*, *Citeseer* and *PubMed*. With only 2 layers, edge enhanced versions behave slightly worse than baseline models. However, with 32 or 64 layers, EEGNNs achieve considerable improvements. This is because the *mis-simplification* applies to all layers. Therefore, the distortion of edge structural information is accumulated from the first to the last layer, resulting in severe performance deterioration.

Table 2: Results on real datasets: mean accuracy (%) ± standard deviation (%)

(a) Number of layers: 2

	Cora	*Citeseer*	*PubMed*	*Texas*	*Wisconsin*	*Cornell*
SGC	**77.01±0.34**	69.18±0.35	75.46±0.28	56.16±4.99	48.59±6.59	57.84±2.76
EE-SGC	76.78±0.29	**69.60±0.37**	**75.80±0.21**	**61.24±6.48**	**53.45±8.95**	**58.92±3.55**
APPNP	**82.22±0.39**	**71.73±0.76**	**79.41±0.48**	61.41±5.27	52.55±7.44	57.73±2.74
EE-APPNP	81.48±0.47	71.45±0.54	78.90±0.52	**66.80±3.74**	**66.23±2.93**	**60.17±6.00**
GCNII	**82.21±0.67**	67.65±0.96	**77.91±1.71**	61.35±8.18	**72.51±4.91**	74.22±8.75
EE-GCNII	81.94±0.51	**81.48±0.59**	77.30±0.97	**64.22±9.02**	70.94±6.10	**75.68±9.78**

(b) Number of layers: 16

	Cora	*Citeseer*	*PubMed*	*Texas*	*Wisconsin*	*Cornell*
SGC	**73.11±0.43**	67.79±0.56	70.45±0.17	56.27±4.92	48.63±6.62	57.84±2.76
EE-SGC	73.07±0.34	**68.55±0.35**	**70.60±0.25**	**59.96±6.46**	**50.59±8.72**	**57.84±2.76**
APPNP	**83.70±0.20**	72.51±0.52	**80.42±0.30**	60.76±5.05	53.29±7.09	57.68±2.79
EE-APPNP	83.47±0.66	**73.20±0.92**	77.90±0.36	**66.08±4.62**	**66.08±3.17**	**61.30±7.18**
GCNII	**84.77±0.37**	72.30±0.80	78.60±0.52	66.38±8.69	70.71±2.40	74.49±8.98
EE-GCNII	84.10±0.57	**72.50±1.40**	**78.81±0.66**	**73.30±3.85**	**78.94±4.90**	**75.24±8.08**

(c) Number of layers: 32

	Cora	*Citeseer*	*PubMed*	*Texas*	*Wisconsin*	*Cornell*
SGC	59.94±0.56	66.17±0.50	68.97±0.19	56.16±4.99	48.59±6.59	57.84±2.76
EE-SGC	**66.46±0.83**	**67.68±0.44**	**70.68±0.68**	**59.46±6.16**	**50.59±8.72**	**58.92±3.15**
APPNP	83.55±0.50	72.11±0.64	80.22±0.34	61.57±5.28	52.71±7.34	57.78±2.75
EE-APPNP	**83.79±0.39**	**72.47±0.53**	79.23±0.27	**66.00±4.33**	**66.39±3.09**	**61.84±7.56**
GCNII	85.34±0.53	73.26±0.86	**79.89±0.33**	70.49±5.48	69.06±2.70	74.05±8.56
EE-GCNII	**85.70±0.41**	**73.45±1.40**	79.72±0.43	**75.24±3.72**	**79.37±0.43**	**74.86±7.84**

(d) Number of layers: 64

	Cora	*Citeseer*	*PubMed*	*Texas*	*Wisconsin*	*Cornell*
SGC	25.65±1.93	63.08±0.52	40.98±1.73	56.16±4.99	48.55±6.58	57.84±2.76
EE-SGC	**35.54±1.36**	**65.42±0.17**	**64.28±0.82**	**59.46±6.16**	**50.31±8.42**	**58.92±3.15**
APPNP	83.58±0.49	72.10±0.48	**80.42±0.42**	61.19±5.29	53.06±7.10	57.68±2.74
EE-APPNP	**83.76±0.41**	**72.16±0.65**	79.94±0.22	**66.00±4.08**	**66.63±3.12**	**61.75±7.43**
GCNII	85.46±0.31	**73.44±1.00**	**80.08±0.37**	69.57±5.70	68.63±1.05	73.19±8.83
EE-GCNII	**85.54±0.59**	72.24±1.26	79.93±0.46	**75.62±3.65**	**76.57±3.89**	**73.26±7.39**

4.3 Application in Financial Data

GNN is widely used in the financial industry for the prediction of stock and bond prices (Wang et al., 2021; Sharma and Sharma, 2020; Feng et al., 2022). To evaluate the efficacy of EEGNN in real-world financial data, we conduct an empirical study using EE-SGC to replace SGC in the current literature and then make a comparison. We use the component stocks from the 'FTSE UK 50 index' with high capitalization and complete records between 2016-01-01 and 2017-12-31. We first construct the graph based on the Pearson correlations between stock returns, by connecting two stocks if their correlation is larger than 0.3. As shown in Figure 4, stocks, indicated by nodes are connected according to their pairwise correlation. Then, we build a learning pipeline using a sequential model of a long short-term memory (LSTM) network, SGC/EE-SGC, and a fully-connected layer. The model was trained using the data in 2016 and tested on the data in 2017. Moreover, the historical returns were used as input data, and the mean squared error between the model outputs and the realized next-day returns was used as the loss function. The Long 20% strategy is adopted to build the portfolio as described in Pacreau et al. (2021). For each trading day, we build a long only portfolio consisting of the top 20% stocks according to the predictive returns. The accumulated returns of the portfolio are shown in Figure 5, where the initial portfolio value is set to be \$100. The results show that the portfolio constructed using EEGNN, which achieved better predictive accuracy, had higher returns.

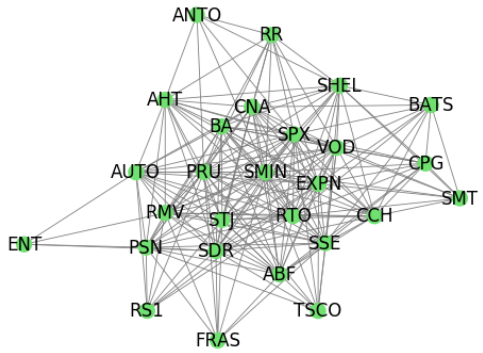


Figure 4: The graph between FTSE UK 50 component stocks. Nodes in green denote individual stocks with their abbreviations in capital letters.

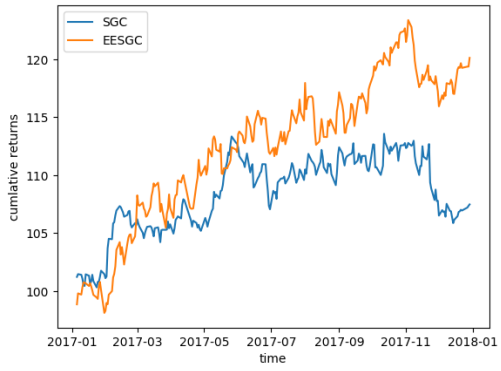


Figure 5: The comparison of cumulative returns using SGC and EE-SGC.

5 CONCLUSION

This paper presents a novel explanation for the performance deterioration of deeper GNNs: *mis-simplification*. We propose DMPGM, a Bayesian nonparametric graph model and its MCMC inference framework. Using the information obtained from DMPGM, we replace the original simple graph by the virtual graph, and use the virtual graph to aggregate the information in the graph. The experiments over various real datasets demonstrate that EEGNN can improve the performance of baseline GNN methods. Our paper paves a new way to use information extracted by statistical graph modelling to improve the performance of GNNs. One limitation of our proposal is that EEGNN only adds the virtual edges to the observed graph without removing edges according to the structural information. It is left for future work to develop a framework that allows to add and remove edges with the structural information simultaneously.

Acknowledgments

We thank the anonymous reviewers for their useful comments during the review process.

Opinions expressed in this paper are those of the authors, and do not necessarily reflect the view of J.P. Morgan. Opinions and estimates constitute our judgement as of the date of this Material, are for informational purposes only and are subject to change without notice. This Material is not the product of J.P. Morgan's Research Department and therefore, has not been prepared in accordance with legal requirements to promote the independence of research, including but not limited to, the prohibition on the dealing ahead of the dissemination of investment research. This Material is not intended as research, a recommendation, advice, offer or solicitation for the purchase or sale of any financial product or service, or to be used in any way for evaluating the merits of participating in any transaction. It is not a research report and is not intended as such. Past performance is not indicative of future results. Please consult your own advisors regarding legal, tax, accounting or any other aspects including suitability implications for your particular circumstances. J.P. Morgan disclaims any responsibility or liability whatsoever for the quality, accuracy or completeness of the information herein, and for any reliance on, or use of this material in any way. Important disclosures at: www.jpmorgan.com/disclosures.

References

Airoldi, E. M., Blei, D. M., Fienberg, S. E., and Xing, E. P. (2008). Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9(65):1981–2014.

Aldous, D. (1997). Brownian excursions, critical random graphs and the multiplicative coalescent. *The Annals of Probability*, 25(2):812–854.

Allamanis, M., Brockschmidt, M., and Khademi, M. (2018). Learning to represent pograms with graphs. In *International Conference on Learning Representations*.

Alon, U. and Yahav, E. (2020). On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*.

Bryant, M. and Sudderth, E. B. (2012). Truly nonparametric online variational inference for hierarchical Dirichlet processes. In *Advances in Neural Information Processing Systems 25*, pages 2699–2707.

Caron, F. and Fox, E. B. (2017). Sparse graphs using exchangeable random measures. *Journal of the Royal Statistical Society: Series B*, 79(5):1295–1366.

Chen, B., Bécigneul, G., Ganea, O.-E., Barzilay, R., and Jaakkola, T. (2021a). Optimal transport graph neural networks. *arXiv:2006.04804*.

Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., and Sun, X. (2020a). Measuring and relieving the over-smoothing

problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3438–3445.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. (2020b). Simple and deep graph convolutional networks. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1725–1735. PMLR.

Chen, T., Zhou, K., Duan, K., Zheng, W., Wang, P., Hu, X., and Wang, Z. (2021b). Bag of tricks for training deeper graph neural networks: a comprehensive benchmark study. *arXiv:2108.10521*.

Cohen, A. C. (1960). Estimating the parameter in a conditional poisson distribution. *Biometrics*, 16(2):203.

Cui, S., Yu, B., Liu, T., Zhang, Z., Wang, X., and Shi, J. (2020). Edge-enhanced graph convolution networks for event detection with syntactic relation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2329–2339.

Feng, S., Xu, C., Zuo, Y., Chen, G., Lin, F., and XiaHou, J. (2022). Relation-aware Dynamic Attributed Graph Attention Network for Stocks Recommendation. *Pattern Recognition*, 121:108119.

Ferguson, T. S. (1973). A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230.

Ghosal, S. and Van der Vaart, A. (2017). *Fundamentals of Nonparametric Bayesian Inference*. Cambridge University Press, Cambridge.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272.

Gong, L. and Cheng, Q. (2019). Exploiting edge features for graph neural networks. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 9211–9219.

Gori, M., Monfardini, G., and Scarselli, F. (2005). A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30.

Hamilton, W. L. (2020). Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159.

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

Huang, K., Xiao, C., Glass, L. M., Zitnik, M., and Sun, J. (2020). SkipGNN: Predicting molecular interactions with skip-graph networks. *Scientific Reports*, 10(1):1–16.

Huang, W., Rong, Y., Xu, T., Sun, F., and Huang, J. (2021). Tackling over-smoothing for general graph convolutional networks. *arXiv:2008.09864*.

Hwang, E., Thost, V., Dasgupta, S. S., and Ma, T. (2021). Revisiting virtual nodes in graph neural networks for link prediction.

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., and others (2017). Population based training of neural networks. *arXiv:1711.09846*.

Ji, S., Pan, S., Cambria, E., Marttinen, P., and Philip, S. Y. (2021). A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514.

Kingman, J. F. C. (1993). *Poisson Processes*. Clarendon Press, Oxford.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Klicpera, J., Bojchevski, A., and Günnemann, S. (2019). Predict then propagate: Graph neural networks meet personalized PageRank. In *International Conference on Learning Representations*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25.

Kroese, D. P., Taimre, T., and Botev, Z. I. (2011). *Handbook of Monte Carlo Methods*. Wiley.

Li, G., Xiong, C., Thabet, A., and Ghanem, B. (2020). DeeperGCN: All you need to train deeper GCNs. *arXiv:2006.07739*.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. S. (2016). Gated graph sequence neural networks. In *International Conference on Learning Representations*.

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

Pacreau, G., Lezmi, E., and Xu, J. (2021). Graph neural networks for asset management. *SSRN Scholarly Paper*.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. (2020). Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*.

Ricci, F. Z., Guindani, M., and Sudderth, E. B. (2022). Thinned random measures for sparse graphs with overlapping communities. In *Advances In Neural Information Processing systems*.

Rong, Y., Huang, W., Xu, T., and Huang, J. (2019). DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *International Conference on Learning Representations*.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Shafie, T. (2015). A multigraph approach to social network analysis. *Journal of Social Structure*, 16(1):1–21.

Sharma, S. and Sharma, R. (2020). Forecasting Transactional Amount in Bitcoin Network Using Temporal GNN Approach. In *International Conference on Advances in Social Networks Analysis and Mining*, pages 478–485.

Singh, A., Huang, Q., Huang, S. L., Bhalerao, O., He, H., Lim, S.-N., and Benson, A. R. (2021). Edge proposal sets for link prediction. *arXiv:2106.15810*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

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

Todeschini, A., Miscouridou, X., and Caron, F. (2020). Exchangeable random measures for sparse and modular graphs with overlapping communities. *Journal of the Royal Statistical Society: Series B*, 82(2):487–520.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations*.

Wang, J., Zhang, S., Xiao, Y., and Song, R. (2021). A Review on graph neural network methods in financial applications. *arXiv:2111.15367*.

Williamson, S. A. (2016). Nonparametric network models for link prediction. *Journal of Machine Learning Research*, 17(202):1–21.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. (2019). Simplifying graph convolutional networks. In *International Conference on Machine Learning*, pages 6861–6871.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24.

Xu, F., Yao, Q., Hui, P., and Li, Y. (2021). Automorphic wquivalence-aware graph neural network. In *Advances in Neural Information Processing Systems*, volume 34.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019). How powerful are graph neural networks? In *International Conference on Learning Representations*.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. (2018). Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pages 5453–5462.

Yang, Z., Cohen, W., and Salakhudinov, R. (2016). Revisiting semi-supervised learning with graph embeddings. In *International conference on Machine Learning*, pages 40–48. PMLR.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. (2020). Graph contrastive learning with augmentations. In *Advances in Neural Information Processing Systems*, volume 33, pages 5812–5823.

Zhang, W., Sheng, Z., Jiang, Y., Xia, Y., Gao, J., Yang, Z., and Cui, B. (2021). Evaluating deep graph neural networks. *arXiv:2108.00955*.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81.

Zhou, M. (2015). Infinite edge partition models for overlapping community detection and link prediction. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, pages 1135–1143.

Zhou, R., Xie, Z., Wan, J., Zhang, J., Liao, Y., and Liu, Q. (2022). Attention and edge-label guided graph convolutional networks for named entity recognition. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6499–6510.

Supplementary Material for 'EEGNN: Edge Enhanced Graph Neural Network with a Bayesian Nonparametric Graph Model'

This supplementary material contains a short review of completely random measures in Appendix A, the details of MCMC steps in Appendix B, technical proofs and derivations in Appendix C, the results of MCMC for several datasets in Appendix D, hyperparameter settings in Appendix E, and computational complexity analysis and code in Appendix F.

A Completely Random Measure

Completely random measures (Ghosal and Van der Vaart, 2017), including gamma process, inverse Gaussian process and stable process, are commonly used as priors for infinite-dimensional latent variables in Bayesian nonparametric models, because their realizations are atomic measures with countable-dimensional supports. Suppose that $(\Omega, \mathscr{F})$ is a Polish sample space, Θ is the set of all bounded measures on $(\Omega, \mathscr{F})$ and $\mathscr{M}$ is a σ-algebra on Θ. A complete random measure from $(\Theta, \mathscr{M})$ into $(\Omega, \mathscr{F})$ can be characterized by its Laplace transform (Kingman, 1993),

$$\mathrm{E}\big[e^{-tP(A)}\big] = \exp\Big\{ -\int_A \int_{(0,\infty]} (1 - e^{-t\pi}) v^c(dx, ds)\Big\},$$

where A is any measurable subset of Ω and $v^c(dx, ds)$ is called the intensity measure. If $v^c(dx, ds) = \kappa(dx)v(ds)$, where $\kappa(\cdot)$ and $v(\cdot)$ are measures on Ω and $(0, \infty]$, respectively, the completely random measure is homogeneous and $v(\cdot)$ is called the Lévy measure. If $\int_0^\infty v(ds) = \infty$, the complete random measure is finite activity.

We can view this completely random measure as a Poisson process on the product space $\Omega \times (0, \infty]$ using the intensity measure and denote this completely random measure as CRM(κ, v). For example, the gamma process $\Gamma\mathrm{P}(\kappa)$ has Lévy measure $v(ds) = s^{-1}e^{-s}ds$ such that $Q(A) \sim \Gamma(\kappa(A), 1)$ if $Q \sim \Gamma\mathrm{P}(\kappa)$, where $\Gamma(\alpha, \beta)$ is a gamma distribution with density $\frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1}e^{-\beta x}$. Therefore, its normalization, Dirichlet process $P \sim \mathrm{DP}(\kappa)$ (Ferguson, 1973) satisfies

$$\big(P(A_1), \ldots, P(A_n)\big) \sim \mathrm{Dirichlet}\big(\kappa(A_1), \ldots, \kappa(A_n)\big)$$

for any partition $\Omega = (A_1, \ldots, A_n)$, where $\bigcup_{i=1}^n A_i = \Omega$ and $A_i \bigcap A_j = \varnothing$ for any i and j. Griffiths–Engen–McCloskey (GEM) distribution, which is the distribution of the weights in a Dirichlet process. For $(\pi_1, \pi_2, \cdots) \sim \mathrm{GEM}(\alpha)$, it can be sampled by $\pi_i = g_i \prod_{l=1}^{i-1} g_l$, where $g_i \sim \mathrm{Beta}(1, \alpha)$ independently (Ghosal and Van der Vaart, 2017).

B MCMC technical details and derivations

B.1 Derivations for Step 1

With the setup of DMPGM in (2) and the formula of moments for Dirichlet-multinomial distribution, we obtain that

$$p(w_{0,1}, \ldots, w_{0,|V|} \mid \bar{w}_0, \boldsymbol{z}, \boldsymbol{c}) \propto \prod_{k=1}^K \frac{\Gamma(\bar{w}_0)}{\Gamma(\bar{w}_0 + n_k)} \prod_{i=0}^N \frac{\Gamma(w_{0,i} + n_{k,i})}{\Gamma(w_{0,i})} \cdot \prod_{i=1}^N v(w_{0,i}) \cdot u\big(\bar{w}_0 - \sum_{i=1}^{|V|} w_{0,i}\big),$$

where $n_{k,i} = \sum_{j=1}^{|V|} \sum_{l=1}^{z_{ij}} \{\mathbb{1}_{c_{ijl}=k} + \mathbb{1}_{c_{jil}=k}\}$, $v(\cdot)$ is the weight intensity measure for the complete random measure of W_0, and $u(\cdot)$ is the density function for $W_0(\Omega)$ that can be derived using its Laplace transform. To infer the posterior distributions for these parameters, we present the gradient of the log-posterior with respect to w_0, which will be used in Hamiltonian Monte Carlo,

$$\nabla_{w_{0,i}} \log p(w_{0,1}, \ldots, w_{0,|V|} \mid \bar{w}_0, \boldsymbol{z}, \boldsymbol{c}) = \sum_{i=1}^{|V|} \sum_{k=1}^K \big\{\Phi(n_{k,i} + w_{0,i}) - \Phi(w_{0,i})\big\}$$
$$+ \sum_{i=1}^{|V|} \nabla_{w_{0,i}} \log v(w_{0,i}) + \nabla_{w_{0,i}} \log u(\bar{w}_0 - \sum_{i=1}^{|V|} w_{0,i}),$$

where Φ is the digamma function.

B.2 Derivations for Step 6

By the formulas of the densities for Poisson distribution and gamma distribution, we have that

$$p(\overline{w_k} \mid \bar{w}_0, \boldsymbol{\pi}, \boldsymbol{c}, \boldsymbol{z}) \propto \frac{(\pi_k \bar{w}_k^2)^{n_k} e^{-\pi_k \bar{w}_k^2}}{n_k!} \cdot \frac{1}{\Gamma(\bar{w}_0)} \bar{w}_k^{\bar{w}_0 - 1} e^{-\bar{w}_k}.$$

Therefore, the log-posterior with respect to $\bar{w}_k$ is

$$\log p(\bar{w}_k \mid \bar{w}_0, \boldsymbol{\pi}, \boldsymbol{c}, \boldsymbol{z}) = (2n_k + \bar{w}_0 - 1) \log \bar{w}_k - \bar{w}_k - \bar{w}_k^2 \pi_k + \text{constant}.$$

Similarly, following Caron and Fox (2017) and Liu et al. (2022), we obtain that

$$p(\bar{w}_0 \mid \bar{w}_k, \boldsymbol{\pi}, \boldsymbol{c}, \boldsymbol{z}) \propto \prod_{k=1}^{K} \frac{1}{\Gamma(\bar{w}_0)} \bar{w}_k^{\bar{w}_0 - 1} e^{-\bar{w}_k} \cdot u(\bar{w}_0).$$

and hence the corresponding log-posterior is

$$\log p(\bar{w}_0 \mid \bar{w}_k, \boldsymbol{\pi}, \boldsymbol{c}, \boldsymbol{z}) = \log u(\bar{w}_0) + \bar{w}_0 \sum_{k=1}^{K} \log(\bar{w}_k) - K \log \Gamma(\bar{w}_0) + \text{constant}.$$

C Technical Proofs and Derivations

C.1 Proof of Theorem 1

The proof for Theorems 3 and 4 in Caron and Fox (2017) can be directly adapted to DMPGM, Therefore, we only provide a sketch of the proof. First, we show that Theorem 3 in Caron and Fox (2017) also holds for DMPGM. We use

$$\tilde{D}_{ij} \mid \{W_k\} \sim \text{Poisson}\big(\sum_k \pi_k W_k([i-1, i]) W([j-1, j])\big),$$

to replace (54) in Appendix C.2 of Caron and Fox (2017). Consequently, (55) holds because for any k we have

$$W_k([0, \alpha])/W_0([0, \alpha]) = O(1) \text{ almost surely as } \alpha \to \infty. \tag{B.1}$$

Second, we show that Theorem 4 in Caron and Fox (2017) also holds for DMPGM. Specifically, (59) becomes

$$X_n \mid \{W_k^{(2)}\} \sim \text{Poisson}\Big[\frac{1}{2} \psi\big\{W(\mathcal{S}_n^{(2)})\big\}\Big],$$

so that (62) in (Caron and Fox, 2017) can be achieved by (B.1). Finally, we complete the proof of Theorem 1 for DMPGM by keeping the remaining parts of the proof of Theorem 4 in Caron and Fox (2017) unchanged.

D Inference results for DMPGM

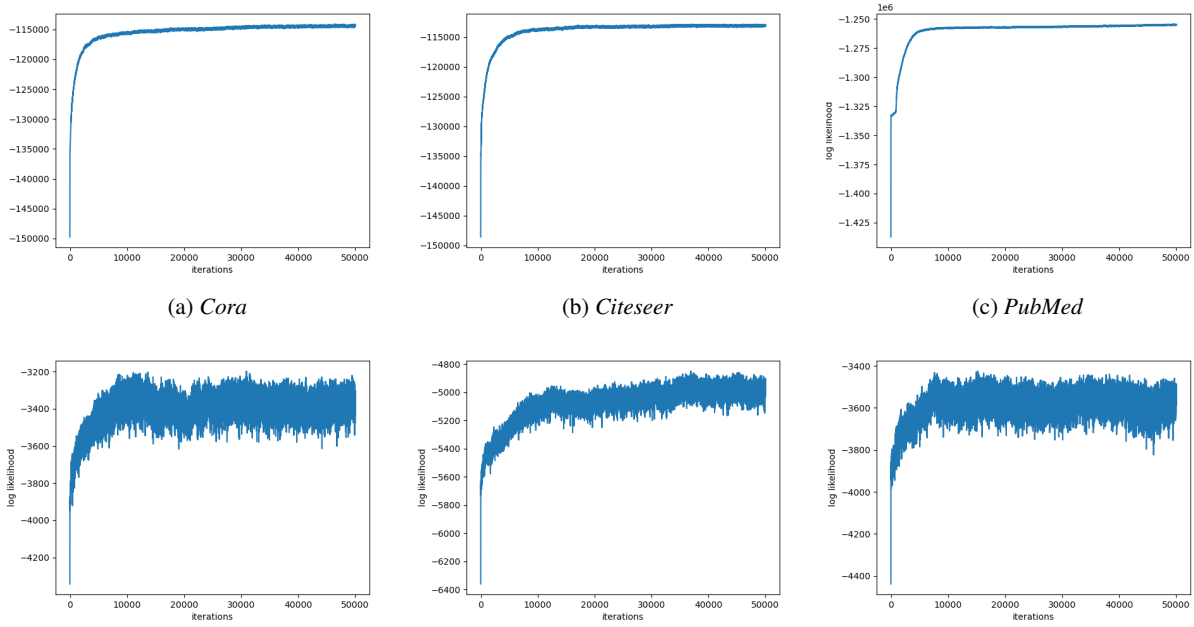
The log likelihood and the number of edge clusters in the training process are shown in Figure 6 and Figure 7, respectively. The inferred edge multiplicity is shown in Figure 8.

E Hyperparameters

We list the hyperparameters used in our experiments in Table 3 below.

F Data and Code

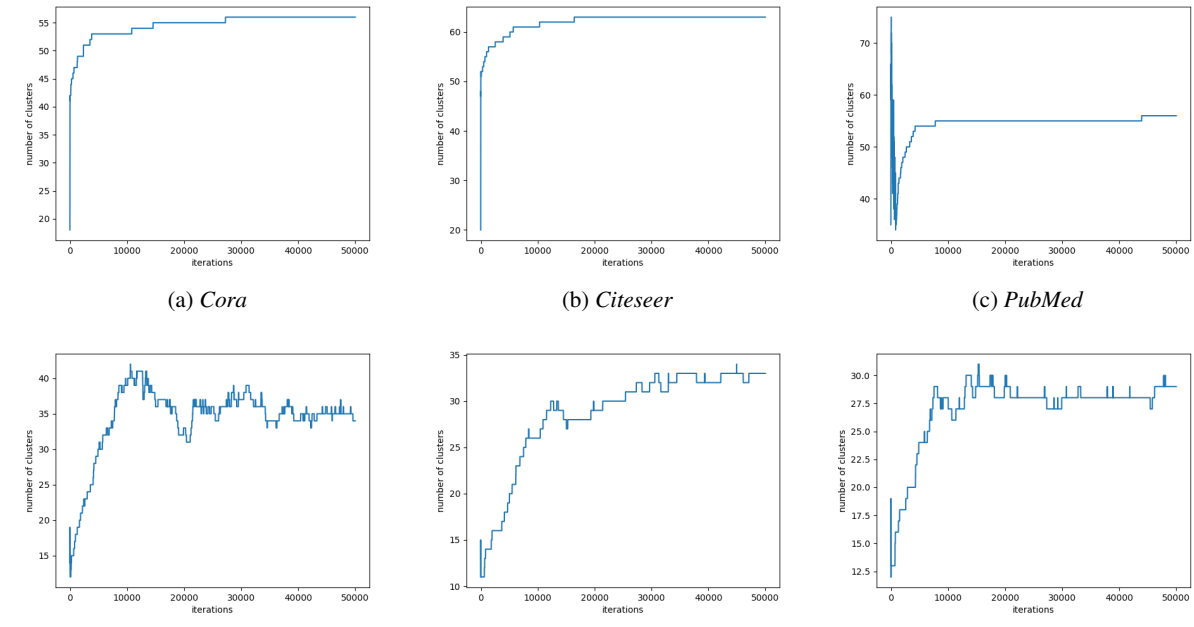
We obtained the datasets from the publically available source `https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html`. All data do not contain personally identifiable information or offensive content. We conducted our experiments on a c5d.4xlarge instance on the AWS EC2 platform, with 16 vCPUs and 32 GB RAM. The codes for training conventional GNNs are from `https://github.com/VITA-Group/Deep_GCN_Benchmarking` under MIT license.



(a) *Cora* (b) *Citeseer* (c) *PubMed*

(d) *Texas* (e) *Wisconsin* (f) *Cornell*

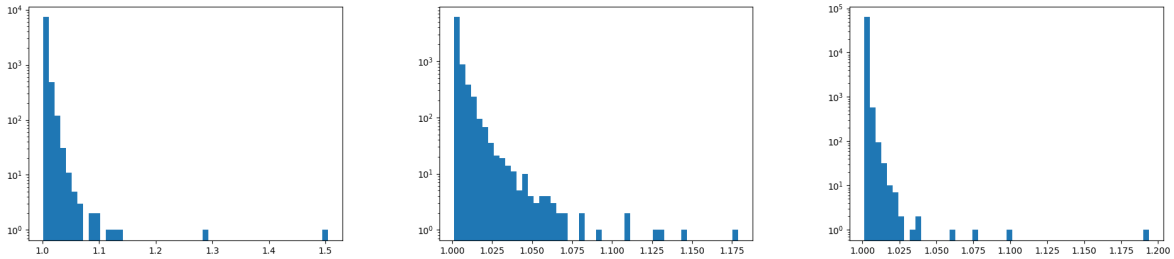
Figure 6: Log-likelihood over the course of the MCMC chain for each dataset.



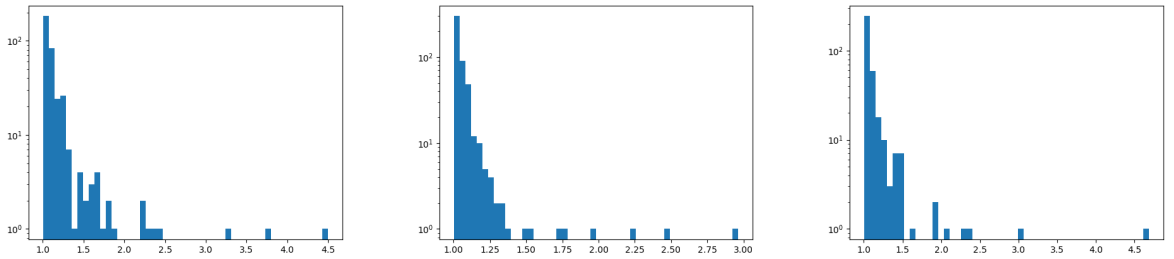
(a) *Cora* (b) *Citeseer* (c) *PubMed*

(d) *Texas* (e) *Wisconsin* (f) *Cornell*

Figure 7: Number of clusters inferred by DMPGM over the course of the MCMC chain.



(a) *Cora* (b) *Citeseer* (c) *PubMed*



(d) *Texas* (e) *Wisconsin* (f) *Cornell*

Figure 8: Histograms of the expected multiplicity of virtual edges formed in the EEGNN framework using each dataset.

	Cora	*Citeseer*	*PubMed*	*Texas*	*Wisconsin*	*Cornell*
dim_hidden	64	256	256	64	64	64
alpha	0.1	0.1	0.1	0.1	0.1	0.1
weight_decay	0.0005	0.0005	0.0005	0.0001	0.0005	0.0005
lr	0.01	0.01	0.01	0.01	0.01	0.01
dropout	0.6	0.7	0.6	0.5	0.5	0.5

(a) Hyperparameters for SGC and EE-SGC.

	Cora	*Citeseer*	*PubMed*	*Texas*	*Wisconsin*	*Cornell*
dim_hidden	64	64	64	64	64	64
alpha	0.1	0.1	0.1	0.1	0.1	0.1
lr	0.01	0.01	0.01	0.01	0.01	0.01
dropout	0.	0.	0.	0.	0.	0.
weight_decay1	0.005	0.005	0.005	0.005	0.005	0.005
weight_decay2	0.	0.	0.	0.	0.	0.
embedding_dropout	0.5	0.5	0.5	0.5	0.5	0.5

(b) Hyperparameters for APPNP and EE-APPNP.

	Cora	*Citeseer*	*PubMed*	*Texas*	*Wisconsin*	*Cornell*
dim_hidden	64	256	256	64	64	64
alpha	0.1	0.1	0.1	0.5	0.5	0.5
lamda	0.5	0.6	0.4	1.5	1.0	1.0
weight_decay1	0.01	0.01	0.0005	0.0001	0.0005	0.0001
weight_decay2	0.0005	0.0005	0.0005	0.0001	0.0005	0.0001
lr	0.01	0.01	0.01	0.01	0.01	0.01
dropout	0.6	0.7	0.6	0.5	0.5	0.5

(c) Hyperparameters for GCNII and EE-GCNII.

Table 3: Hyperparameters in the training.

Chapter 4

Bayesian Nonparametric for Sequential Data

This chapter is dedicated to an article that has been submitted and is currently under review. The article can be accessed online at `https://arxiv.org/abs/2305.14543`.

DF^{2}M: An Explainable Deep Bayesian Nonparametric Model for High-Dimensional Functional Time Series

Yirui Liu[1] **Xinghao Qiao**[1] **Yulong Pei**[2] **Liying Wang**[3]
[1]London School of Economics and Political Science
[2]Eindhoven University of Technology [3]University of Liverpool

Abstract

In this paper, we present Deep Functional Factor Model (DF^{2}M), a Bayesian nonparametric model for analyzing high-dimensional functional time series. The DF^{2}M makes use of the Indian Buffet Process and the multi-task Gaussian Process with a deep kernel function to capture non-Markovian and nonlinear temporal dynamics. Unlike many black-box deep learning models, the DF^{2}M provides an explainable way to use neural networks by constructing a factor model and incorporating deep neural networks within the kernel function. Additionally, we develop a computationally efficient variational inference algorithm for inferring the DF^{2}M. Empirical results from four real-world datasets demonstrate that the DF^{2}M offers better explainability and superior predictive accuracy compared to conventional deep learning models for high-dimensional functional time series.

1 Introduction

Functional time series, which refers to a sequential collection of functional objects exhibiting temporal dependence, has attracted increasing attention in recent years. With the advent of new data collection technology and enhanced computational power, high-dimensional datasets containing a large collection of functional time series are becoming increasingly available. Examples include annual age-specific mortality rates across different countries, daily energy consumption curves from different households, and cumulative intraday return trajectories for hundreds of stocks, to list a few. Those data can be represented as a p-dimensional functional time series $\boldsymbol{Y}_t(\cdot) = \big(Y_{t1}(\cdot), \ldots, Y_{tp}(\cdot)\big)^T$, $t = 1, \ldots, n$, where each $Y_{tj}(\cdot)$ is a random function defined on a compact interval $\mathcal{U}$ and 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 poses a challenging task, as it requires not only dimension reduction techniques to solve the high-dimensional problem, but also functional approaches to handle the infinite-dimensional nature of the curve data, as well as time series modeling to capture the temporal dependence structure. Several statistical methods have been proposed to address these issues, as exemplified by [1, 2, 3]. However, these approaches often assume the existence of a linear and Markovian dynamic over time, which may fail to characterize the complex nonlinear or non-Markovian temporal dependence that is commonly encountered in practical real-world scenarios.

On the other hand, although deep learning has achieved attractive results in computer vision and natural language processing (NLP) [4, 5, 6, 7], the direct application of deep neural networks to handle high-dimensional functional time series is rather difficult. For time series data, one major problem is that deep neural networks are black-box models and lack explainability, making it hard to understand the cross-sectionally and serially correlated relationships. However, explainability is essential in many applications. For example, in finance, healthcare, and weather forecasting, the accuracy and reliability of the model's predictions have considerable impacts on business decisions, patient outcomes, or people's safety, respectively. Additionally, the non-stationarity of the data and the large number of parameters in deep neural networks pose extra challenges for training.

In this paper, we present an explainable approach, deep functional factor model (DF^{2}M), with the ability to discover nonlinear and non-Markovian dynamics in high-dimensional functional time series. As a Bayesian nonparametric model, DF^{2}M uses a functional version of factor model to perform dimension reduction, incorporates an Indian buffet process prior in the infinite-dimensional loading matrix to encourage column sparsity [8], utilizes a functional version of Gaussian process dynamical model to capture temporal dependence within latent functional factors, and adopts deep neural networks to construct the temporal kernel.

DF^{2}M enjoys several advantages for analyzing high-dimensional functional time series. First, by representing observed curve variables using a smaller set of latent functional factors, it allows for a more intuitive understanding of the underlying structure in data, hence enhancing model explainability. As a structural approach, DF^{2}M not only offers a clear and interpretable mapping of relationships between variables, but also provides a built-in guard against overfitting [9]. As a result, predictions can be interpreted and understood in a meaningful way, which is critical for decision-making and subsequent analysis. Second, DF^{2}M can discover non-Markovian and nonlinear temporal dependence in the functional latent factor space, and hence has the potential to predict future values more accurately. Finally, DF^{2}M provides a flexible framework that combines modern sequential deep neural networks with a backbone Bayesian model, allowing for the use of sequential deep learning techniques such as gated recurrent unit (GRU) [10], long short-term memory (LSTM) [11] and attention mechanisms [6].

2 Preliminaries

2.1 Indian Buffet Process

The Indian buffet process (IBP) [12] 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 [13]. 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 $\kappa(u, v) = \exp\big(-\frac{|u-v|^2}{2\ell^2}\big)$ and the Ornstein–Uhlenbeck kernel $\kappa(u, v) = \exp\big(-\frac{|u-v|}{\ell}\big)$, where ℓ is the length-scale parameter. Additionally, the kernel function can be made more complex using the kernel trick [14] 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) [15] can be employed to model vector-valued random fields. It is defined as $\boldsymbol{X}(\cdot) = \big(X_1(\cdot), \ldots, X_M(\cdot)\big)^T$, 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

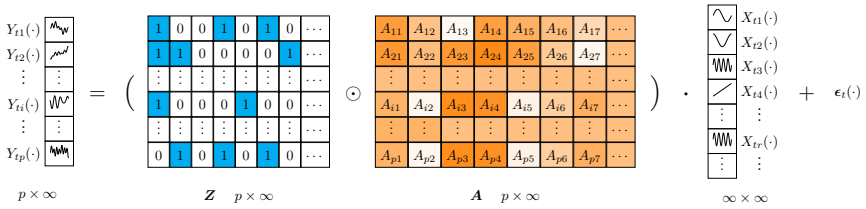


Figure 1: Sparse functional factor model. For $\boldsymbol{Z}$, the blue (or white) cells represent 1 (or 0). For $\boldsymbol{A}$, the darker (or lighter) shades of orange indicate larger (or smaller) values.

matrix encoding the similarities between pairs of tasks. The MTGP can effectively capture inter-task correlations and improve predictions [16].

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 [17]. 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 $\boldsymbol{x}_t = g_{dec}(\boldsymbol{h}_t)$ and $\boldsymbol{h}_t = g_{enc}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_{t-1})$, where $\boldsymbol{h}_t$ 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 $\boldsymbol{h}_t = \text{LSTM}(\boldsymbol{x}_{1:t})$ and $\boldsymbol{h}_t = \text{GRU}(\boldsymbol{x}_{1:t})$, respectively, where $\boldsymbol{x}_{1:t} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t)$. Moreover, attention mechanisms, 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 $\boldsymbol{h}_t = \sum_{i=1}^{t-1} \omega(\boldsymbol{k}_t, \boldsymbol{q}_\tau)\boldsymbol{\nu}_{t-\tau}$, where key $\boldsymbol{k}_t$, query $\boldsymbol{q}_t$, and value $\boldsymbol{\nu}_t$ are intermediate representations generated by linear or nonlinear transformations of $\boldsymbol{x}_t$. We denote such attention mechanisms as $\boldsymbol{h}_t = \text{ATTN}(\boldsymbol{x}_{1:t})$. 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,

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

where $\boldsymbol{Y}_t(\cdot)$ is the observed functional time series, $\boldsymbol{Z}$ is a binary matrix sampled from the Indian buffet process, $\boldsymbol{Z} \sim \text{IBP}(\alpha)$, Hadamard (elementwise) product is represented by $\odot$, $\boldsymbol{A}$ is the loading weight matrix with $A_{tr} \sim \text{Normal}(0, \sigma_A^2)$ for any $r \in \mathbb{N}^+$ independently; $\boldsymbol{X}_t(\cdot) = \big(X_{t1}(\cdot), X_{t2}(\cdot), \ldots, X_{tr}(\cdot), \ldots\big)^T$ is a set of latent functional factor time series, $\boldsymbol{\epsilon}_t(\cdot)$ is the idiosyncratic component with a Gaussian distributed white noise process on a scale σ_ϵ. In particular, we do not specify the number of latent factors. Instead, $\boldsymbol{Y}_t(\cdot)$, $\boldsymbol{Z}$ and $\boldsymbol{A}$ can be regarded as a $p \times \infty$ matrices, and $\boldsymbol{X}_t(\cdot)$ is 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, so we do not need to specify a fixed number for the dimension of the factor space. This nonparametric approach introduces flexibility and provides a foundation for inferring the number of factors in the posterior distribution, using the nonparametric inference framework, such as Gibbs sampling [18], online variational inference [19], merge-split algorithm [20] and conditional and adaptively truncated variational inference [21].

Moreover, the Indian buffet process can also provide column sparsity [22] to $\boldsymbol{Z}$ and hence also to the loading matrix $\boldsymbol{Z} \odot \boldsymbol{A}$. This means that most of the elements in each row are zeros, because w_k in the stick-breaking representation of the IBP goes to zero as k increases. In the factor model, this column sparsity means that each factor has an impact on only a small fraction of functional variables [8], equivalently that the factors are related to each other through a hierarchy [12].

3.2 Gaussian Process Dynamical Model for Functional Data

Second, by projecting high-dimensional observations $\boldsymbol{Y}_t(\cdot)$ into low-dimensional latent functional factors $\boldsymbol{X}_t(\cdot)$, the sequential structure of the time series model can be captured via the factors. To model such 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 $u, v \in \mathcal{U}$,

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

where t and s indicate two time stamps, $\kappa_{\mathcal{X}}$ and $\kappa_{\mathcal{U}}$ are the kernels defined on $\mathcal{X}$ and $\mathcal{U}$, respectively. $\mathbb{I}(r = l)$ is an indicator function that takes on the value of 1 if $r = l$ and 0 otherwise. $\kappa_{\mathcal{X}}$ is with respect to historical information on different periods and can be regarded as a temporal kernel. In parallel, $\kappa_{\mathcal{U}}$ can be regarded as a spatial kernel. Therefore, $\boldsymbol{X}_r(\cdot) = \big(X_{1r}(\cdot), \ldots, X_{tr}(\cdot), \ldots, X_{nr}(\cdot)\big)$ belongs a multi-task Gaussian process [15] such that for any $u_1, \ldots, u_L \in \mathcal{U}$, $\mathrm{vec}\big(\boldsymbol{X}_r(u_1, \ldots, u_L)\big) \sim \mathrm{Normal}(\boldsymbol{0}, \boldsymbol{\Sigma}_{\mathcal{X}} \otimes \boldsymbol{\Sigma}^u_{\mathcal{U}})$, where $\otimes$ denotes the Kronecker product, or equivalently $\boldsymbol{X}_r(u_1, \ldots, u_L) \sim \mathrm{MatrixNormal}(\boldsymbol{0}, \boldsymbol{\Sigma}^u_{\mathcal{U}}, \boldsymbol{\Sigma}_{\mathcal{X}})$ using the matrix normal distribution [23], where $\boldsymbol{X}_r(u_1, \ldots, u_L) = \big[X_{tr}(u_j)\big]_{1\le t\le n, 1\le j\le L}$, $\boldsymbol{\Sigma}_{\mathcal{X}} = \big[\kappa_{\mathcal{X}}(\mathcal{X}_t, \mathcal{X}_s)\big]_{0\le t,s\le n-1}$, and $\boldsymbol{\Sigma}^u_{\mathcal{U}} = \big[\kappa_{\mathcal{U}}(u_i, u_j)\big]_{1\le i,j\le L}$. In Appendix B, we demonstrate the detailed relationship between the multi-task Gaussian process and the matrix normal distribution. In the context of literature, the n-task Gaussian process is used to refer to the multiple outputs generated by the model, each of which corresponds to a specific timestamp in our time series setting. For the convenience of expression, we denote the presented multi-task Gaussian process as

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

It is worth noting that the latent factors $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_r, \ldots$ are not mutually independent, because they share the temporal kernel $\kappa_{\mathcal{X}}$ with respect to $\mathcal{X}_{t-1}$ that contains common historical information across factors till period $t-1$. Therefore, the multi-task Gaussian process can measure the similarity and dependence across periods. Moreover, in our setting, the temporal kernel used for prediction depends solely on past information $\mathcal{X}_{t-1}$ rather than current information $\mathcal{X}_t$. This approach allows for a forward-looking prediction framework based on historical data only.

The proposed model can be regarded as a functional version of the Gaussian process dynamical model [24]. See their connections in Appendix C. There are several advantages of using the proposed model to capture the temporal dependence of functional time series. First, the temporal kernel and spatial kernel are separable, providing us with a closed form and computational convenience. Moreover, since the temporal kernel is related to the entire historical information instead of the latest state only, the model can be non-Markovian. For example, we can define the temporal kernel as $\kappa(\mathcal{X}_{t-1}, \mathcal{X}_{s-1}) = \alpha_1 \int \boldsymbol{X}_{t-1}(u)^T \boldsymbol{X}_{s-1}(u)du + \alpha_2 \int \boldsymbol{X}_{t-2}(u)^T \boldsymbol{X}_{s-2}(u)du$, to incorporate features from the last two periods. Finally, using the kernel trick, we can introduce nonlinearity by setting a nonlinear kernel function. This paves the way for us to construct deep kernels, which will be discussed in Section 3.3.

3.3 Deep Temporal Kernels

Finally, to capture the complicated nonlinear latent temporal structure, we employ deep neural networks to construct the kernel function. However, in order to use deep kernels for functional time series, we need two extra steps compared to standard deep kernels [25, 26, 27, 28, 29, 30]. (i) As $\boldsymbol{X}_t(\cdot)$ 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, where $\mathcal{F}$ is 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 [31, 32]), adaptive functional neural network [33], or even simply $\boldsymbol{X}_t(u_0, \ldots, u_L)$ with $u_0, \ldots, u_L \in \mathcal{U}$. (ii) The d-dimensional vectors serve as inputs for deep neural networks, and the outputs generated by these networks are then employed to construct kernel functions. Specifically, we first transform the input vector as

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

where $\boldsymbol{X}_{t-1} = \big(X_{t-1,1}, \ldots, X_{t-1,r} \ldots\big)^T$, F is the mapping function, and H is 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 demonstrated their effectiveness in modeling complex patterns and dependencies. Since the inputs for the temporal kernels are ordered sequences from $\mathcal{X}_0$ to $\mathcal{X}_{n-1}$, we should use unidirectional deep neural networks instead of bidirectional networks. We then use the transformed representations $\boldsymbol{h}_t$ and $\boldsymbol{h}_s$ to construct a kernel function

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

where $\kappa(\cdot, \cdot)$ is a suitable kernel function, such as the squared exponential kernel or the Ornstein–Uhlenbeck kernel. We note that the temporal kernel is related to the historical values of all the relevant factors and shared across factors.

In summary, using the functional version of the sparse factor model, sequential deep learning kernel, and Gaussian process dynamical model presented above, we define a probabilistic generative model for high-dimensional functional time series and name it as deep functional factor model, DF^{2}M. To mitigate overfitting, one can apply spectrum normalization, which effectively instills a Lipschitz condition into the neural networks according to [34]. This suggests that the inputs for κ could reflect the distance between $\mathcal{X}_{t-1}$ and $\mathcal{X}_{s-1}$.

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 [35]. 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 [36]. 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 [37]. 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,

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

where $\boldsymbol{v} = (v_1, \ldots, v_K)^T$ with $v_1, \ldots, v_K \in \mathcal{U}$ and K is the number of inducing points. We construct the variational distribution for inducing variables as $q(X_{tr}(\boldsymbol{v})) = \text{Normal}(\boldsymbol{\mu}_{tr}, \boldsymbol{S}_{tr})$. It is noteworthy that the conditional prior distribution for $X_r(\cdot)$, i.e., the first term on the right-hand-side of equation (7), cannot be factorized as $\prod_{t=1}^{n} p\big(X_{tr}(\cdot) \mid X_{tr}(\boldsymbol{v})\big)$, because they have temporal dependence. However, benefiting from the setting of equation (7), the conditional prior distribution appears both in variational and prior distributions and hence 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

$$2\mathrm{KL}\Big[q\big(\boldsymbol{X}_r(\boldsymbol{v})\big) \parallel p\big(\boldsymbol{X}_r(\boldsymbol{v}) \mid \kappa_{\mathcal{X}}, \kappa_{\mathcal{U}}\big)\Big] = \mathrm{trace}\Big((\boldsymbol{\Sigma}_{\mathcal{X}}^{-1} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1})\big(\boldsymbol{S}_r + \mathrm{vec}(\boldsymbol{\mu}_r)\mathrm{vec}(\boldsymbol{\mu}_r)^T\big)\Big) + K \log|\boldsymbol{\Sigma}_{\mathcal{X}}| + n \log|\boldsymbol{\Sigma}_{\mathcal{U}}^{vv}| - \sum_{t=1}^{n} \log|\boldsymbol{S}_{tr}| - nK, \tag{9}$$

where $\boldsymbol{\mu}_r = (\boldsymbol{\mu}_{1r}, \ldots, \boldsymbol{\mu}_{nr})$, $\boldsymbol{S}_r = \mathrm{diag}(\boldsymbol{S}_{1r}, \ldots, \boldsymbol{S}_{nr})$, and $\boldsymbol{\Sigma}_{\mathcal{U}}^{vv} = \big[\kappa_{\mathcal{U}}(v_j, v_j)\big]_{1 \le i,j \le K}$. 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 [38, 35, 39] 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, though this distribution is Gaussian conditional on $\boldsymbol{X}_r(\boldsymbol{v})$, directly sampling from a $nL \times nL$ matrix is computationally expensive. To address this issue, we take advantage of the separability of the temporal and spatial kernels as described in Section 3.2, and propose the following method to accelerate the computation of the ELBO. 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 $\begin{bmatrix} \boldsymbol{\Sigma}_{\mathcal{U}}^{uu} & \boldsymbol{\Sigma}_{\mathcal{U}}^{uv} \\ \boldsymbol{\Sigma}_{\mathcal{U}}^{uv\,T} & \boldsymbol{\Sigma}_{\mathcal{U}}^{vv} \end{bmatrix}$, where $\boldsymbol{\Sigma}_{\mathcal{U}}^{uu} = \big[\kappa_{\mathcal{U}}(u_i, u_j)\big]_{1 \le i,j \le L}$, and $\boldsymbol{\Sigma}_{\mathcal{U}}^{uv} = \big[\kappa_{\mathcal{U}}(u_i, v_j)\big]_{1 \le i \le L, 1 \le j \le K}$.

Proposition 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 $\boldsymbol{u}$*

$$\mathrm{E}\left(X_{tr}(\boldsymbol{u})\right) = \boldsymbol{\Sigma}_{\mathcal{U}}^{uv}\boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1}\boldsymbol{\mu}_{tr}. \tag{10}$$

It means that for MTGP, the variational mean is independent of the inducing variables at timestamps other than the current one. See also a similar proposition for Gaussian process regression in [15].

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

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

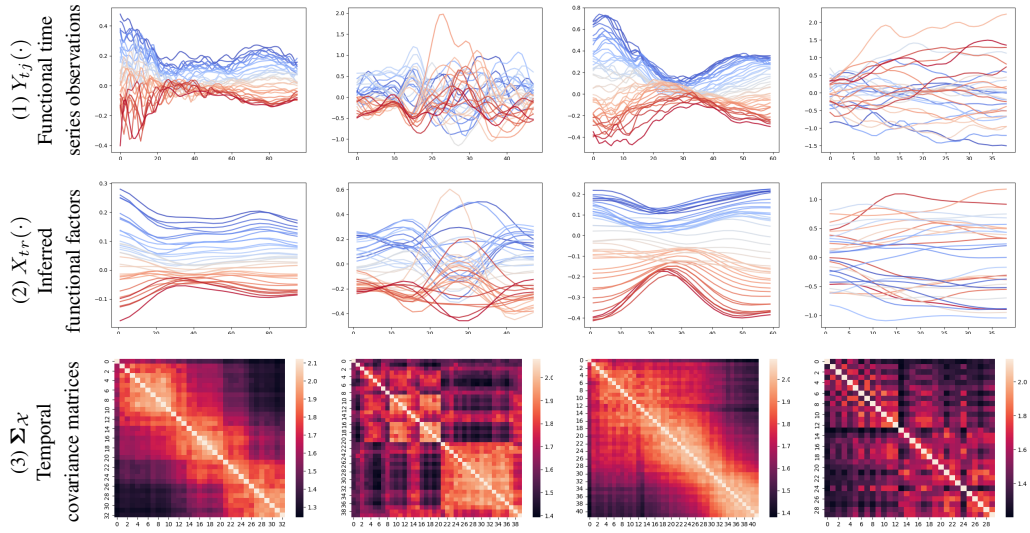
The first part is solely dependent on the variational variance of $\boldsymbol{X}_{tr}(\boldsymbol{v})$, and the second part is independent of the variational distributions of all inducing variables. In particular, the first part corresponds to a group of independent Gaussian processes such that $\tilde{\boldsymbol{X}}_{tr}^{(1)}(\boldsymbol{u}) \sim \mathrm{Normal}(\boldsymbol{\Sigma}_{\mathcal{U}}^{uv}\boldsymbol{\Sigma}_{\mathcal{X}}^{vv\,-1}\boldsymbol{\mu}_{tr}, \boldsymbol{\Sigma}_{\mathcal{U}}^{uv}\boldsymbol{\Sigma}_{\mathcal{X}}^{vv\,-1}\boldsymbol{S}_{tr})$ for any $\boldsymbol{u}$, and $\tilde{\boldsymbol{X}}_r^{(2)}(\cdot)$ is a zero-mean multi-task Gaussian process such that $\tilde{\boldsymbol{X}}_r^{(2)}(\boldsymbol{u}) \sim \mathrm{MatrixNormal}(\boldsymbol{0}, \boldsymbol{\Sigma}_{\mathcal{U}}^{uu} - \boldsymbol{\Sigma}_{\mathcal{U}}^{uv}\boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1}\boldsymbol{\Sigma}_{\mathcal{U}}^{uv\,T}, \boldsymbol{\Sigma}_{\mathcal{X}})$ for any $\boldsymbol{u}$. Therefore, benefiting from Propositions 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. $\tilde{\boldsymbol{X}}_r^{(1)}(\cdot)$ can be efficiently sampled because it only depends on inducting variables at the same period instead of all periods.

Proposition 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 alter the variational mean. Moreover, the corresponding ELBO of DF^{2}M in equation (8) is modified only 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, for any matrix $\mathbf{M} = (M_{ij})$, we denote its Frobenius norm by $\|\mathbf{M}\|_F = \big(\sum_{i,j} M_{ij}^2\big)^{\frac{1}{2}}$.

See Appendices D.3, D.4 and D.5 for the derivations of Proposition 1, 2 and 3, respectively. Based on these propositions, 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 an efficient way of computing the ELBO compared to direct sampling, which necessitates the complete Cholesky decomposition of the $nL \times nL$ matrix.



(a) *Japanese Mortality* (b) *Energy Consumption* (c) *Global Mortality* (d) *Stock Intraday*

Figure 2: A visualization of real datasets in the experiments. Rows 1 and 2 use a blue-to-red gradient to denote time progression, with blue for older and red for recent data. Row 3 employs brightness variations to represent covariance, with brighter areas indicating higher covariance.

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 $\mathbf{\Sigma}_{\mathcal{X}}$, 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 $\{\tau_j^1, \tau_j^2\}_{1\leq j\leq M}$ and $\{m_{tj}\}_{1\leq t\leq n, 1\leq j\leq M}$ for India buffet process $\boldsymbol{Z}$, $\{\eta_{tj}, \sigma_{tj}^A\}_{1\leq t\leq n, 1\leq j\leq M}$ for loading weight matrix $\boldsymbol{A}$, the idiosyncratic noise scale σ_ϵ, and the parameters in spatial kernel $\kappa_{\mathcal{U}}(\cdot,\cdot)$. In this step, the gradient of ELBO is accelerated by sampling $X_{tr}(\cdot)$ independently according to Proposition 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 $\mathbf{\Sigma}_{\mathcal{X}}$. Although any mapping function F can be used in our model, it is natural to choose $F\big(\boldsymbol{X}_t(\cdot)\big) = \boldsymbol{X}_t(\boldsymbol{v})$, such that there is no 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 [36].

Once we have observed the data at time n, we use the trained model to generate a posterior distribution, which 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:

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

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 $\mathbf{\Sigma}_{\mathcal{X}}^{n+1,1:n}$ is a $1\times n$ matrix given by $\mathbf{\Sigma}_{\mathcal{X}}^{n+1,1:n} = \big[\kappa_{\mathcal{X}}(\mathcal{X}_{n+1},\mathcal{X}_0),\cdots,\kappa_{\mathcal{X}}(\mathcal{X}_{n+1},\mathcal{X}_n)\big]$. See Appendix D.6 for the derivations. By repeating this process iteratively, we can generate a sequence of predictions for future time steps, 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 [40]. For the preprocessing, each dataset is cleaned and transformed into an appropriate format for analysis. See Appendix F for the details. We denote the data as $\{Y_{tj}(u_k)\}_{1\le t\le n, 1\le j\le p, 1\le k\le K}$, where K is the number of observations per curve. We plot examples of functional time series for a randomly selected j in Row 1 of Figure 2.

5.2 Experiment Setup and Metrics

Moreover, 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, and then repeatedly move the training window by one period, refit the model, and make the h-step-ahead prediction. We compute the mean absolute prediction error (MAPE) and mean squared prediction error (MSPE) by

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

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, and denote them as DF^{2}M-LIN, DF^{2}M-LSTM, DF^{2}M-GRU and G-ATTN, respectively. For DF^{2}M, the outputs of 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. The proposed inference algorithm is used to infer DF^{2}M and make predictions.

5.3 Empirical Results

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 ($\mathbf{\Sigma}_{\mathcal{X}}$) 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 *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 relatively smaller, as people are on holiday.

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 *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	MAPE	MSPE	MAPE	MSPE	MAPE	MSPE	MAPE	MSPE	MAPE	MSPE	MAPE	MSPE	MAPE	MSPE	MAPE
Japanese Mortality	1	4.707	1.539	7.808	2.092	3.753	1.205	4.989	1.447	4.092	1.318	8.800	1.691	**3.608**	**1.119**	13.44	3.166
	2	4.567	1.446	8.774	2.227	4.164	1.322	5.597	1.523	4.395	1.402	8.552	1.809	**3.839**	**1.203**	14.85	3.363
	3	5.623	1.635	9.228	2.313	4.513	1.427	6.501	1.684	4.898	1.537	10.41	1.865	**3.985**	**1.264**	16.17	3.546
Energy Consm.	1	10.29	2.334	16.16	2.939	**8.928**	**2.176**	13.51	2.635	9.132	2.204	15.55	2.872	14.22	2.741	17.03	3.130
	2	17.58	3.060	18.95	3.214	**11.60**	**2.478**	19.71	3.278	15.49	2.801	24.02	3.518	18.70	3.141	17.79	3.216
	3	17.64	3.100	20.27	3.342	17.26	3.063	24.61	3.759	**14.13**	**2.801**	23.91	3.626	19.03	3.163	18.24	3.268
Global Mortality	1	10.78	2.319	16.84	2.783	**7.672**	**1.726**	13.28	2.332	8.741	1.967	14.12	2.211	9.905	2.141	39.52	5.332
	2	9.300	2.041	18.05	2.949	**8.088**	**1.823**	16.29	2.572	8.714	1.951	15.33	2.403	10.46	2.237	41.83	5.506
	3	9.706	2.106	19.93	3.174	**8.954**	**1.978**	17.08	2.680	9.730	2.110	17.53	2.597	11.12	2.324	43.95	5.643
Stock Intraday	1	**99.58**	**6.424**	137.5	7.896	107.5	6.741	193.3	9.281	102.5	6.675	414.0	14.12	104.2	6.695	103.4	6.579
	2	101.2	6.505	127.8	7.491	118.8	7.141	176.0	9.283	117.3	7.339	445.9	14.66	103.4	6.646	**98.39**	**6.392**
	3	**89.82**	**6.269**	139.1	7.924	113.6	7.294	213.8	10.20	95.49	6.649	427.2	14.07	93.93	6.427	91.21	6.275

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, [8] develop a finite-dimensional functional factor model for dimension reduction, while [2] carry out autocovariance-based dimension reduction, and [41] adopt segmentation transformation. However, all these methods use either a vector autoregressive model (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, [25, 26, 27, 28, 29, 30] use deep kernels in the Gaussian process for classification or regression tasks. Differently, we pioneer a framework that employs a deep kernel specifically for time series prediction. Lastly, [42, 24, 43] adopt MTGPs to model cross-sectional correlations among static data. Contrarily, we apply a factor model to describe cross-sectional relationships, and the temporal kernel is constructed based on the features of 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 way. A potential limitation of our study lies in our reliance on simple spatial kernels, neglecting to account for the intricate relationships within the observation space $\mathcal{U}$. We leave this for future work.

References

[1] Yuan Gao, Han Lin Shang, and Yanrong Yang. High-dimensional functional time series forecasting: An application to age-specific mortality rates. *Journal of Multivariate Analysis*, 170:232–243, 2019.

[2] Jinyuan Chang, Cheng Chen, Xinghao Qiao, and Qiwei Yao. An autocovariance-based learning framework for high-dimensional functional time series. *Journal of Econometrics*, 2023.

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

[4] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016. titleTranslation:.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, June 2016.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[7] Amirsina Torfi, Rouzbeh A Shirvani, Yaser Keneshloo, Nader Tavaf, and Edward A Fox. Natural language processing advancements by deep learning: A survey. *arXiv preprint arXiv:2003.01200*, 2020.

[8] Shaojun Guo, Xinghao Qiao, and Qingsong Wang. Factor modelling for high-dimensional functional time series. *arXiv:2112.13651*, 2021.

[9] Christoph Molnar. *Interpretable Machine Learning*. 2022.

[10] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. Publisher: MIT press.

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

[13] Christopher K Williams and Carl Edward Rasmussen. *Gaussian Processes for Machine Learning*. MIT Press Cambridge, 2006.

[14] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, 36(3):1171–1220, 2008.

[15] Edwin V Bonilla, Kian Chai, and Christopher Williams. Multi-task gaussian process prediction. In *Advances in Neural Information Processing Systems*, volume 20, 2007.

[16] Pablo Moreno-Muñoz, Antonio Artés, and Mauricio Álvarez. Heterogeneous multi-output gaussian process prediction. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

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

[18] Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[19] Chong Wang and David M. Blei. Truncation-free online variational inference for Bayesian nonparametric models. In *Advances in Neural Information Processing Systems 25*, 2012.

[20] Michael C Hughes and Erik Sudderth. Memoized online variational inference for Dirichlet process mixture models. In *Advances in Neural Information Processing Systems 26*, pages 1133–1141, 2013.

[21] Yirui Liu, Xinghao Qiao, and Jessica Lam. CATVI: Conditional and adaptively truncated variational inference for hierarchical bayesian nonparametric models. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, pages 3647–3662, 2022.

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

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

[24] Jack Wang, Aaron Hertzmann, and David J. Fleet. Gaussian process dynamical models. In *Advances in Neural Information Processing Systems*, volume 18, 2005.

[25] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 370–378, 2016.

[26] Maruan Al-Shedivat, Andrew Gordon Wilson, Yunus Saatchi, Zhiting Hu, and Eric P. Xing. Learning scalable deep kernels with recurrent structure. *The Journal of Machine Learning Research*, 18(1):2850–2886, 2017.

[27] Hui Xue, Zheng-Fan Wu, and Wei-Xiang Sun. Deep spectral kernel learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 4019–4025, 2019.

[28] Wenliang Li, Danica J. Sutherland, Heiko Strathmann, and Arthur Gretton. Learning deep kernels for exponential family densities. In *International Conference on Machine Learning*, pages 6737–6746, 2019.

[29] Joe Watson, Jihao Andreas Lin, Pascal Klink, Joni Pajarinen, and Jan Peters. Latent derivative Bayesian last layer networks. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, pages 1198–1206, 2021.

[30] Vincent Fortuin. Priors in Bayesian deep learning: a review. *International Statistical Review*, page 12502, 2022.

[31] Neil Bathia, Qiwei Yao, and Flavio Ziegelmann. Identifying the finite dimensionality of curve time series. *The Annals of Statistics*, 38:3352–3386, 2010.

[32] Siegfried Hormann, Lukasz Kidzinski, and Marc Hallin. Dynamic functional principal components. *Journal of the Royal Statistical Society: Series B*, 77:319–348, 2015.

[33] Junwen Yao, Jonas Mueller, and Jane-Ling Wang. Deep Learning for Functional Data Analysis with Adaptive Basis Layers. In *International Conference on Machine Learning*, pages 11898–11908. PMLR, 2021.

[34] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[35] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: a review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

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

[37] Oliver Hamelijnck, William Wilkinson, Niki Loppi, Arno Solin, and Theodoros Damoulas. Spatio-temporal variational Gaussian processes. In *Advances in Neural Information Processing Systems*, volume 34, pages 23621–23633, 2021.

[38] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M Blei. Automatic differentiation variational inference. *Journal of machine learning research*, 2017.

[39] Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Artificial Intelligence and Statistics*, pages 814–822, 2014.

[40] Lajos Horváth, Piotr Kokoszka, and Gregory Rice. Testing stationarity of functional time series. *Journal of Econometrics*, 179(1):66–82, 2014.

[41] Jinyuan Chang, Qin Fang, Xinghao Qiao, and Qiwei Yao. On the Modelling and Prediction of High-Dimensional Functional Time Series.

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

[43] Michalis Titsias and Neil D. Lawrence. Bayesian Gaussian process latent variable model. In *Proceedings of the 13th international conference on artificial intelligence and statistics*, pages 844–851, 2010.

Appendix

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 [11], 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,

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

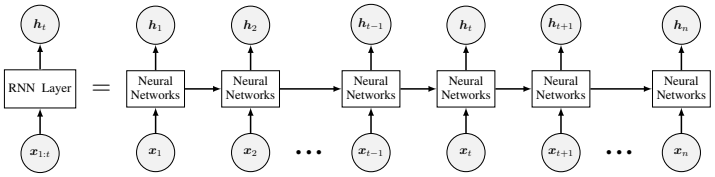
where $\boldsymbol{z}_t$ and $\boldsymbol{r}_t$ are the activation vectors for update gate and reset gate, respectively, and $\tilde{\boldsymbol{h}}_t$ is cell input activation vector.

A.2 Attention Mechanism

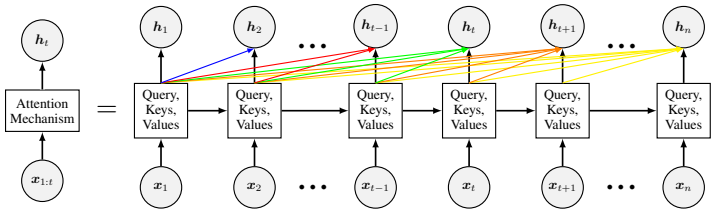
Attention mechanism is a deep learning model that is especially effective for sequential data prediction. It allows the model to assign different weights to different parts of the input, rather than treating them all equally. This can improve the model's ability to make predictions by allowing it to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

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

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.



(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.

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 $\text{vec}\big(\boldsymbol{X}_r(u_1, \ldots, u_L)\big) \sim \text{Normal}(\boldsymbol{0}, \boldsymbol{\Sigma}_{\mathcal{X}} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^u)$,

where

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

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

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

C Functional version of Gaussian Process Dynamical Model

Following [24], 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 $\boldsymbol{a}_i(\cdot) = \{a_{1i}(\cdot), a_{2i}(\cdot), \ldots, a_{ri}(\cdot), \ldots, \}^T$. This functional version of dynamical system corresponds to equation (3) in [24]. 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 $\kappa_{\mathcal{X}}(\mathcal{X}_{t-1}, \mathcal{X}_{s-1}) = \sum_i \langle \phi_i(\mathcal{X}_{t-1}), \phi_i(\mathcal{X}_{s-1}) \rangle$.

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

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

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 Kullback–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 Proposition 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

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

Therefore, $\mathrm{E}_q\Big[\text{vec}\big(\boldsymbol{X}_r(\boldsymbol{u})\big)\Big] = \boldsymbol{\Sigma}_{\mathcal{U}}^{uv}\boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1}\text{vec}(\boldsymbol{\mu}_r)$, 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 Proposition 2

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

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

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 Proposition 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,

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

D.5 Proof for Proposition 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

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

where $\beta_{ir} = (\boldsymbol{Z} \odot \boldsymbol{A})_{ir}$, $\boldsymbol{Y}_i(\boldsymbol{u}) = \begin{bmatrix} Y_{1i}(u_1) & \cdots & Y_{ni}(u_1) \\ \cdots & \cdots & \cdots \\ Y_{1i}(u_L) & \cdots & Y_{ni}(u_L) \end{bmatrix}$, and $\boldsymbol{X}_r(\boldsymbol{u}) = \begin{bmatrix} X_{1r}(u_1) & \cdots & X_{nr}(u_1) \\ \cdots & \cdots & \cdots \\ X_{1r}(u_L) & \cdots & X_{nr}(u_L) \end{bmatrix}$. 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 $\tilde{\boldsymbol{X}}_r^{(1)}(\boldsymbol{u}) = \begin{bmatrix} \tilde{X}_{1r}^{(1)}(u_1) & \cdots & \tilde{X}_{nr}^{(1)}(u_1) \\ \cdots & \cdots & \cdots \\ \tilde{X}_{1r}^{(1)}(u_L) & \cdots & \tilde{X}_{nr}^{(1)}(u_L) \end{bmatrix}$ and $\tilde{\boldsymbol{X}}_r^{(2)}(\boldsymbol{u}) = \begin{bmatrix} \tilde{X}_{1r}^{(2)}(u_1) & \cdots & \tilde{X}_{nr}^{(2)}(u_1) \\ \cdots & \cdots & \cdots \\ \tilde{X}_{1r}^{(2)}(u_L) & \cdots & \tilde{X}_{nr}^{(2)}(u_L) \end{bmatrix}$.

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 Proposition 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.

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).

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 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.

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.

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

The results indicate that the DF^{2}M-based methods exhibit a smaller or comparable standard deviation compared to other competitors.

Chapter 5

Conclusion

In conclusion, this thesis introduces innovative methods within Bayesian nonparametric machine learning to model complex datasets. It investigates three primary areas of application: natural language processing, deep graph neural networks, and high-dimensional functional time series. The methodologies proposed include Conditional and Adaptively Truncated Variational Inference (CATVI) for hierarchical Bayesian nonparametric models, the Edge Enhanced Graph Neural Network (EEGNN) for improving deep graph neural networks, and a Bayesian nonparametric Deep Functional Factor Model (DF^2M) for analysing high-dimensional functional time series.

Overall, the examination of Bayesian nonparametric models alongside deep learning, as depicted in this thesis, shows the potential to tackle challenging issues such as overfitting and high dimensionality. The inherent flexibility of Bayesian nonparametric models, when combined with the predictive power of deep learning, presents a promising approach for modeling complex datasets.

References

J. Choi and K.-E. Kim. Nonparametric Bayesian inverse reinforcement learning for multiple reward functions. In *Advances in Neural Information Processing Systems 25*, 2012.

T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230, 1973.

Z. Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521 (7553):452–459, 2015.

S. Ghosal and A. Van der Vaart. *Fundamentals of Nonparametric Bayesian Inference*. Cambridge University Press, Cambridge, 2017.

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

M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.

D. P. Kroese, T. Taimre, and Z. I. Botev. *Handbook of Monte Carlo Methods*. John Wiley and Sons, 2011.

E. B. Sudderth and M. I. Jordan. Shared segmentation of natural scenes using depen-

dent Pitman–Yor processes. In *Advances in Neural Information Processing Systems 21*, pages 1585–1592, 2009.

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

S. A. Williamson. Nonparametric network models for link prediction. *Journal of Machine Learning Research*, 17(202):1–21, 2016.

M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni. Bayesian nonparametric federated learning of neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 7252–7261, 2019.