Imperial College London

Imperial College London
Department of Computing

Probabilistic Learning and Generation in Deep Sequence Models

Wenlong Chen

December 18, 2025

This thesis is submitted for the degree of *Doctor of Philosophy*

Declaration of Originality

I hereby declare that the work in this thesis is my own. The work of others has been appropriately referenced. A full list of references is given in the bibliography.

Copyright

Acknowledgements

First and foremost, I would like to thank my supervisor, Yingzhen Li, for her unwavering support throughout my PhD. Yingzhen, in many ways, is the best supervisor that I can ever imagine. In addition to her enthusiasm for research, she is also such a knowledgeable, patient, modest, and fun person to work with. I am grateful for the freedom and encouragement that Yingzhen gave me to pursue my own ideas and explore different research directions. Her deep insights and candid feedback always inspired me to think deeper and helped me grow as a better researcher. Thanks to Yingzhen, these four years become one of the most memorable experiences in my life.

I am very lucky to have worked with excellent collaborators during my PhD, from whom I learned a great deal: Yegor Klochkov, Naoki Kiyohara, Harrison Zhu, Wenlin Chen, Lapo Rastrelli, Shavindra Jayasekera, Jacob Si, Filippo Valdettaro, Yohan Jung, Hyungi Lee, Thomas Möllenhoff, Mohammad Emtiyaz Khan, Bolian Li, and Ruqi Zhang. I especially thank Yegor Klochkov and Mohammad Emtiyaz Khan for offering incredible internship experiences at Bytedance AI lab and RIKEN AIP. Yegor introduced me to the world of machine learning fairness and optimization, and Emti encouraged me to think outside the box.

I am grateful to everyone I shared interesting conversations with during my PhD: Carles Balsells Rodas, Zijing Ou, Xavier Sumba Toral, Jiaming Zhang, Wen Wu, Rui Xia, Jean-Francois Ton, Ruocheng Guo, Muhammad Faaiz Taufiq, Zonghao Chen, Yang Liu, Junyu Xuan, Xue Yan, Theodore Papamarkou, Andre Freitas, Danilo S. Carvalho, Hugo Monzón Maldonado, Marco Miani, Luke Ong, and many others. I also want to express my heartfelt thanks to Yingxue Yang, with whom I have shared more than twenty years of friendship, for all the jokes and memorable moments we have shared together in my leisure life.

Many thanks to Felipe Tobar and Arno Solin for examining my thesis. I am very grateful for their time and effort in reading my thesis and providing feedback.

Finally, I owe a lot to my family and loved ones. I would like to express my deepest

gratitude to my parents for their unconditional support throughout this journey.

Abstract

Deep sequence models have achieved profound success across a wide range of data modalities. Despite exceptional predictive performance, the main concern of their deployment centers around the lack of uncertainty awareness. In contrast, probabilistic models quantify the uncertainty associated with unobserved variables with rules of probability. Notably, Bayesian methods leverage Bayes' rule to express our belief of unobserved variables given some observed variables in a principled way. Since exact Bayesian inference is computationally infeasible at scale, approximate inference is required in practice. Two major bottlenecks of Bayesian methods, especially when applied in deep neural networks, are prior specification and approximation quality. In Chapter 3 and 4, we investigate how the architectures of deep sequence models themselves can be informative for specifying priors or choosing approximation methods in probabilistic models. We first develop an approximate Bayesian inference method tailored to the Transformer architecture based on the similarity between attention mechanism and sparse Gaussian process. Next, we exploit the long-range memory preservation capability of HiPPOs (High-order Polynomial Projection Operators) to construct an interdomain inducing point for Gaussian process, which successfully memorizes the history in online or continual learning. In addition to the progress of deep sequence models in predictive tasks, sequential generative models consisting of a sequence of latent variables (e.g., diffusion models), are popularized in the domain of deep generative models. Inspired by the explicit self-supervised signals for these latent variables in diffusion models, in Chapter 5, we explore the possibility of improving other deep generative models with self-supervised signals for their latent states, and investigate desired probabilistic structures over the sequence of latent states in sequential generation. Overall, this thesis leverages inductive biases in deep sequence models to design probabilistic inference or structure, which bridges the gap between deep sequence models and probabilistic models, leading to mutually reinforced improvement.

Contents

1 Introduction **16**

1.1 Thesis Outline and Contributions 18

1.2 List of Publications 20

2 Background **22**

2.1 Deep Sequence Model Architectures 22

2.1.1 Transformer 22

2.1.2 HiPPO: Recurrent Memory with Optimal Polynomial Projections 25

2.2 Scalable Probabilistic Learning 28

2.2.1 Bayesian Inference 28

2.2.2 Variational Inference 29

2.3 Gaussian Processes 31

2.3.1 Exact Inference for Regression 32

2.3.2 Sparse Variational Gaussian Process 34

2.3.3 Deep Kernel Learning and Deep Gaussian Processes 37

2.4 Deep Sequential Generative Models 39

2.4.1 Deep Latent Variable Models 39

2.4.2 Hierarchical Variational Autoencoders 41

2.4.3 Diffusion Models 43

3 Calibrating Transformers via Sparse Gaussian Processes **45**

3.1 Introduction 46

3.2 Sparse Gaussian Process Attention 47

3.2.1 Attention as the Mean of a Sparse Variational Gaussian Process 47

3.2.2 Standard SGPA & its Inefficiency for Self-attention 48

3.3 Improving Time & Memory Efficiencies via Decoupled SGPA 49

3.3.1 Preliminaries of Decoupled SVGPs 50

3.3.2 Decoupled SGPA 51

3.4 Transformer Based on Decoupled SGPA 52

3.5 Experiments 54

3.5.1 In-distribution Calibration . . . 55
3.5.2 Robust Prediction on Out-of-distribution Data . . . 57
3.5.3 Out-of-distribution Detection . . . 59
3.5.4 Summary of Additional Results . . . 61
3.6 Related Work . . . 61
3.7 Conclusion and Future Work . . . 62

4 Recurrent Memory for Online Interdomain Gaussian Processes 63
4.1 Introduction . . . 63
4.2 Preliminaries . . . 65
4.2.1 Interdomain Gaussian Processes . . . 65
4.2.2 Online Sparse Gaussian Processes . . . 66
4.2.3 Gaussian Processes Variational Autoencoders . . . 67
4.3 Interdomain Inducing Point Gaussian Processes with HiPPO . . . 68
4.3.1 HiPPO as Interdomain Inducing Variables . . . 68
4.3.2 Adapting the Kernel Matrices over Time . . . 68
4.4 Extending OHSVGP to Multidimensional Input . . . 71
4.5 Related Work . . . 74
4.6 Experiments . . . 75
4.6.1 Online Time Series Prediction . . . 77
4.6.2 Continual Learning on UCI Datasets . . . 79
4.6.3 Continual Learning for High Dimensional Time Series Prediction 85
4.7 Conclusion . . . 88

5 Your Image is Secretly the Last Frame of a Pseudo Video 89
5.1 Introduction . . . 90
5.2 Motivation . . . 91
5.2.1 Diffusion Model vs Hierarchical VAE . . . 92
5.2.2 Improving Image Generation via Pseudo Video Generation . . . 94
5.3 Improved Reconstruction and Generation in VQ-VAE with Pseudo Videos . . . 94
5.3.1 Preliminaries of VQVAE . . . 95
5.3.2 Experiments . . . 96
5.4 Improved Generation via Higher-order Markov Pseudo Videos . . . 100
5.4.1 Is First-order Markov Chain the Optimal Choice for Creating Pseudo Videos? . . . 100
5.4.2 Experiments . . . 102
5.5 Related Work . . . 105
5.5.1 Sequential Generative Models . . . 105
5.5.2 Self-supervised Learning . . . 106

5.6 Conclusion and Discussions . . . 106

6 Conclusion and Future Work **108**
6.1 Thesis Summary . . . 108
6.2 Limitations and Future Research Directions . . . 110

A Supplementary Material for Chapter 2 **131**
A.1 HiPPO Variants in Addition to HiPPO-LegS . . . 131

B Supplementary Material for Chapter 3 **133**
B.1 Derivations . . . 133
B.1.1 ELBO Derivations for SVGP . . . 133
B.1.2 Derivation of ELBO for Transformers Based on SGPA . . . 134
B.2 Uncertainty Calibration Metrics . . . 136
B.2.1 Proper Scoring Rule . . . 136
B.2.2 Calibration Error . . . 136
B.3 Additional Experiments . . . 137
B.3.1 Empirical Comparison Between Standard SGPA and Decoupled SGPA . . . 137
B.3.2 Graph Property Regression with ZINC Dataset . . . 138
B.4 Detailed Experimental Setups and Hyperparameters . . . 139
B.5 Running Time . . . 141
B.6 Connection with Sparse within Sparse GP . . . 142
B.7 Results in Tables . . . 143

C Supplementary Material for Chapter 4 **151**
C.1 Computing the Prior Inducing Covariance $\mathbf{K}_{\mathbf{uu}}^{(t)}$ as Direct ODE Evolution 151
C.2 Finite Basis Approximation of Posterior OHSVGP . . . 153
C.3 SVGPVAE Model Details . . . 154
C.4 Additional Results . . . 155
C.4.1 Results for Time Series Regression with Trainable Kernel Hyperparameters . . . 155
C.4.2 Comparison of Basis–measure Variants . . . 155

D Supplementary Material for Chapter 5 **157**
D.1 Proof of Theorem 5.4.1 . . . 157
D.2 Hyperparameters . . . 158

List of Figures

2.1 Illustration of one head (h) of multi-head self attention in one layer of Transformer. . . . 24
2.2 An illustration of HiPPO-LegS approximation of a toy time series (c) $y(x)$. Here x is used to denote arbitrary time index. (a) Time-dependent basis functions with end time index $t = T/2$ and $t = T$ ($T = 1$ here). (b) Evolution of the memory state $\boldsymbol{c}(t)$. (d) Finite basis approximation $\hat{y}^t(x)$ with the end time index $t = T/2$ and T. 25
2.3 The graphical model of prediction of sparse variational Gaussian process (SVGP). . . . 35
2.4 The graphical model of prediction of an L-layer deep Gaussian process with sparse GP approximations. . . . 39
2.5 Graphical model of a deep latent variable model. . . . 40
2.6 Graphical model of a deep sequential generative variable model. . . . 41

3.1 Illustration of one head (h) of multi-head self attention in one layer of (a) vanilla Transformer, (b) Transformer based on standard SGPA and (c) Transformer based on decoupled SGPA. . . . 48
3.2 Test set accuracy (or MCC for CoLA) & calibration metrics of Transformers or ViTs trained on CIFAR10 (1st row), CIFAR100 (2nd row), IMDB (3rd row) and CoLA (4th row). . . . 56
3.3 Test set accuracy & calibration metrics of ViTs trained on CIFAR10 (top row) and CIFAR100 (bottom row) with data augmentation. . . . 56
3.4 Test set MCC & calibration metrics for OOD test set of Transformers trained on COLA. . . . 57
3.5 Test set accuracy & calibration metrics on CIFAR10-C (top row) and CIFAR100-C (bottom row) against skew intensity of corruption for ViTs trained on corresponding clean data without data augmentation. 57
3.6 Test set accuracy & calibration metrics on CIFAR10-C (top row) and CIFAR100-C (bottom row) against skew intensity of corruption for ViTs trained on corresponding clean data with data augmentation. . . 58

3.7 AUROC (top) and AUPR (bottom) metrics for OOD detection using ViTs trained on CIFAR10 without data augmentation. 59
3.8 AUROC (top) and AUPR (bottom) metrics for OOD detection using ViTs trained on CIFAR100 without data augmentation. 59
3.9 AUROC (top) and AUPR (bottom) metrics for OOD detection using ViTs trained on CIFAR10 with data augmentation. 60
3.10 AUROC (top) and AUPR (bottom) metrics for OOD detection using ViTs trained on CIFAR100 with data augmentation. 60

4.1 Online HiPPO Sparse Variational Gaussian Process (OHSVGP) on a toy time series with 2 tasks. Here x is used to denote arbitrary time index. **(a)** Time-dependent basis functions with end time index $x = t_1$ and $x = t_2$. **(b)** Evolution of optimal approximate posterior of inducing variables (mean ± 2 marginal standard deviation). **(c)**, **(d)**, **(e)** illustrate predictive mean ± 2 standard deviation of posterior online GP, OHSVGP and finite basis reconstruction of posterior OHSVGP, respectively. 71
4.2 Decision boundaries of OSVGP, and OHSVGP models with different sorting criteria after each task (3 in total) on the Two-moon dataset. For OSVGP, we visualize the inducing locations with red color. 72
4.3 Test set NLPD over the learned tasks vs. number of learned tasks for Solar Irradiance and Audio signal prediction dataset. 77
4.4 Test set RMSE over the learned tasks vs. number of learned tasks for Solar Irradiance and Audio signal prediction dataset. 77
4.5 Predictive mean ± 2 standard deviation of OSGPR, OVC, OVFF, and OHSGPR after task 10 of the Solar dataset. M = 50 inducing variables are used. 77
4.6 Test set NLPD per task after continually learning each task for all the 5 tasks on COVID dataset. 78
4.7 Test set ECE per task after continually learning each task for all the 5 tasks on COVID dataset. 79
4.8 Test set NLPD per task after continually learning each task for all the 10 tasks on UCI Skillcraft dataset. 81
4.9 Test set RMSE per task after continually learning each task for all the 10 tasks on UCI Skillcraft dataset. 82
4.10 Test set NLPD per task after continually learning each task for all the 10 tasks on UCI Powerplant dataset. 83
4.11 Test set RMSE per task after continually learning each task for all the 10 tasks on UCI Powerplant dataset. 84

4.12 Test set NLPD per task after continually learning each task for all the 10 tasks on ERA5 dataset. 86
4.13 Test set RMSE per task after continually learning each task for all the 10 tasks on ERA5 dataset. 87

5.1 Diffusion model vs HVAE: compared to standard HVAEs, diffusion models incorporate inductive bias for its intermediate latent states with self-supervised signals. 92
5.2 Generated digits from HVAE with encoder fixed according to the heat equation and standard HVAE with learnable encoder. Both HVAEs use the same decoder architecture as in Rissanen et al. (2023) 93
5.3 An example of pseudo video constructed by transforming an image of a dog using blurring. 97
5.4 Ground-truth images and reconstructed images from VQVAE/CViViT trained on CIFAR10. 98
5.5 Ground-truth images and reconstructed images from VQVAE/CViViT trained on CelebA. 98
5.6 Examples of a pseudo video constructed by adding Gaussian noise to a CIFAR10 image using first-order Markov chain (top) and high-order Markov chain (bottom). 103
5.7 Generated images from the video diffusion models trained on 4-frame high-order Markov pseudo videos of CIFAR10 and CelebA, respectively. 104

B.1 Test set regression error and NLL metrics for Transformers trained on ZINC. 138
B.2 AUROC and AUPR metrics for OOD detection using Transformers trained on ZINC. 138

C.1 Test set RMSE over the learned tasks vs. number of learned tasks for Solar Irradiance and Audio signal prediction dataset (keep updating kernel hyperparameters). 155
C.2 Test set NLPD over the learned tasks vs. number of learned tasks for Solar Irradiance and Audio signal prediction dataset (keep updating kernel hyperparameters). 155
C.3 Comparison of OHSGPR based on different HiPPO variants on a toy online regression dataset. 156

List of Tables

3.1 Complexity comparison for standard and decoupled SGPA. 52
3.2 Average ranks of different methods in terms of AUROC and AUPR over 6 OOD detection tasks 58

4.1 Wall-clock accumulated runtime for learning all the tasks on a single NVIDIA RTX3090 GPU in seconds, of different models for time series prediction experiments. 79

5.1 Inception Score (IS) of generated digits from HVAE with encoder fixed according to the heat equation and standard HVAE with learnable encoder. 93
5.2 Last-frame FID/PSNR of images produced by C-ViViT (reconstruction), VideoGPT (AR generation) and Phenaki (latent masked generation) trained on pseudo videos constructed from CIFAR10 and CelebA images. 1-frame results are obtained from their image counterparts VQVAE (reconstruction), ImageGPT (AR generation) and MaskGit (latent masked generation) trained on original CIFAR10 and CelebA images. 99
5.3 Last-frame FID of images generated by video diffusion models trained on pseudo videos constructed from CIFAR10 and CelebA images (with both first-order Markov or high-order Markov Gaussian noise data augmentation). 1-frame results are obtained from an image diffusion model trained on the original CIFAR10 and CelebA images with equivalent UNet architecture. 103
5.4 Last-frame FID of images generated by video diffusion models trained on pseudo videos constructed from CIFAR10 and CelebA images with high order Markov Gaussian noise data augmentation. 1-frame results are obtained from an image diffusion model trained on the original CIFAR10 and CelebA images. Here, the 1-frame models use 4,000 diffusion steps, while the 4-frame models use 3,000 diffusion steps overall. 105

5.5 Last-frame FID of images generated by video diffusion models trained on pseudo videos constructed from CIFAR10 images with high-order Markov data augmentation (either Gaussian noise or Gaussian blur). 105

B.1 Test set accuracy and NLL of ViTs based on standard and two variants of decoupled SGPA, trained on CIFAR10 without data augmentation. 137

B.2 The computational time (in s) for a single batch at the inference stage with 10 Monte Carlo samples for CoLA (batch size = 227) and CIFAR10 (batch size = 200) (results obtained using a single Nvidia GTX 2080 Ti GPU card). 141

B.3 The training time (in s) of one epoch for SGPA and MLE on CoLA (batch size = 32) and CIFAR10 (batch size = 100) (results obtained using a single Nvidia GTX 2080 Ti GPU card). 142

B.4 In-distribution performance: sentiment analysis with IMDB 143

B.5 In-distribution performance: linguistic acceptability with CoLA . . . 143

B.6 In-distribution performance: image classification for CIFAR10 without data augmentation 144

B.7 In-distribution performance: image classification for CIFAR10 with data augmentation 144

B.8 In-distribution performance: image classification for CIFAR100 without data augmentation 144

B.9 In-distribution performance: image classification for CIFAR100 with data augmentation 145

B.10 In-distribution performance: graph property regression with ZINC dataset 145

B.11 OOD robustness: linguistic acceptability with CoLA dataset 145

B.12 Accuracy of CIFAR10-C for ViTs trained on clean data without data augmentation. 145

B.13 Accuracy of CIFAR10-C for ViTs trained on clean data with data augmentation. 146

B.14 Accuracy of CIFAR100-C for ViTs trained on clean data without data augmentation. 146

B.15 Accuracy of CIFAR100-C for ViTs trained on clean data with data augmentation. 146

B.16 NLL of CIFAR10-C for ViTs trained on clean data without data augmentation. 146

B.17 NLL of CIFAR10-C for ViTs trained on clean data with data augmentation. 147

B.18 NLL of CIFAR100-C for ViTs trained on clean data without data augmentation. . . . 147
B.19 NLL of CIFAR100-C for ViTs trained on clean data with data augmentation. . . . 147
B.20 ECE of CIFAR10-C for ViTs trained on clean data without data augmentation. . . . 147
B.21 ECE of CIFAR10-C for ViTs trained on clean data with data augmentation. . . . 148
B.22 ECE of CIFAR100-C for ViTs trained on clean data without data augmentation. . . . 148
B.23 ECE of CIFAR100-C for ViTs trained on clean data with data augmentation. . . . 148
B.24 MCE of CIFAR10-C for ViTs trained on clean data without data augmentation. . . . 148
B.25 MCE of CIFAR10-C for ViTs trained on clean data with data augmentation. . . . 149
B.26 MCE of CIFAR100-C for ViTs trained on clean data without data augmentation. . . . 149
B.27 MCE of CIFAR100-C for ViTs trained on clean data with data augmentation. . . . 149
B.28 AUROC and AUPR metrics for OOD detection using ViTs trained on CIFAR10 without data augmentation. . . . 149
B.29 AUROC and AUPR metrics for OOD detection using ViTs trained on CIFAR10 with data augmentation. . . . 150
B.30 AUROC and AUPR metrics for OOD detection using ViTs trained on CIFAR100 without data augmentation. . . . 150
B.31 AUROC and AUPR metrics for OOD detection using ViTs trained on CIFAR100 with data augmentation. . . . 150
B.32 AUROC and AUPR metrics for OOD detection using Transformers trained on ZINC. . . . 150

D.1 Hyperparamters used for C-ViViT architecture and optimizer. . . . 159
D.2 Hyperparamters used for VideoGPT architecture and optimizer. . . . 159
D.3 Hyperparamters used for Phenaki architecture and optimizer. . . . 159
D.4 Hyperparamters used for UNet architecture and optimizer for Video diffusion. . . . 160

Chapter 1

Introduction

While deep learning techniques can be traced back to the last century (McCulloch and Pitts, 1943; Rosenblatt, 1958), their revival began in the early 2010s, with the advent of a series of breakthroughs (Krizhevsky et al., 2012; Graves et al., 2013b; Brown et al., 2020), proving their effectiveness in modeling complex functions and scalability to large datasets (LeCun and Hinton, 2015). Given that input features from many data modalities are in the form of sequences (e.g., natural language, audio, and video) or can be represented by sequences (e.g., an image can be viewed as a sequence of patches (Dosovitskiy et al., 2021)), recent progress in deep learning is largely driven by advances in deep sequence models (Vaswani et al., 2017; Gu et al., 2020, 2022; Gu and Dao, 2023), which tend to become a unifying modeling paradigm across different modalities and keep shattering our expectation of their upper limit (Jumper et al., 2021; OpenAI et al., 2024b; Zeni et al., 2024).

One of the key components of deep sequence models is a few carefully designed network architectures tailored to sequence modeling. The success of Transformers (Vaswani et al., 2017) based on attention architecture has been observed in a wide range of fields including computer vision (Dosovitskiy et al., 2021), natural language processing (Devlin et al., 2019; Brown et al., 2020; OpenAI et al., 2024b), speech recognition (OpenAI et al., 2024a), and applications in scientific domains (Jumper et al., 2021; Zeni et al., 2024). In addition, deep state space models (SSMs; (Gu et al., 2020, 2022; Gu and Dao, 2023)), initially designed to efficiently model long-term dependencies in sequences with extensive context, offer a competitive alternative to Transformers and have been deployed in a growing number of applications (Zhu et al., 2024; Li et al., 2024; Quan and Li, 2024).

In the deep generative modeling domain, sequential generative models (Sønderby et al., 2016; Ho et al., 2020) extend single latent variable models by incorporating a sequence

of latent variables which improve the model expressiveness, and instead of generating data in the target domain from noise in just one shot, these models gradually transform noise into target data with multiple steps. Among them, diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021b) and their variants (Kingma et al., 2021; Nichol and Dhariwal, 2021; Song et al., 2021a; Rissanen et al., 2023; Bansal et al., 2023; Hoogeboom and Salimans, 2023) have shown impressive performance in generating photorealistic images, and these models have also been adapted to other data modalities in addition to vision (Benita et al., 2023; Campbell et al., 2023; Lou et al., 2024).

Although deep sequence models have become foundational building blocks in modern machine learning systems, it is important to be aware that they are still not the ultimate answer to all applications. For example, these models are often unaware of the limits of their own knowledge and provide unreliable uncertainty in their predictions, which prevents their deployment in safety-critical applications (Xiong et al., 2024; Wen et al., 2024).

On the other hand, probabilistic methods, based on the law of probability and Bayesian statistics (Bayes, 1763), provide a principled framework to reason about uncertainty and study the structure of random variables. In this framework, we start with a prior distribution representing our initial belief of the unknown, and the knowledge from the observed data is used to update the prior into a posterior distribution, typically with uncertainty reduction, via Bayes' rule. Furthermore, the framework is well suited for sequential learning or decision making under uncertainty. The Bayes' update can be carried out whenever new data arrives by treating the old posterior as the new prior.

Clever integration of the complementary strength of probabilistic methods into deep sequence models may address some fundamental problems associated with them, including unreliable uncertainty estimate. In the past decade, many efforts have been made to exploit probabilistic methods to improve the reliability of deep learning, which gives rise to a vital research area, namely Bayesian deep learning (Wilson and Izmailov, 2022; Arbel et al., 2023; Papamarkou et al., 2024; Chen et al., 2025c). However, two main bottlenecks of Bayesian statistics are amplified when applied in deep learning. First, meaningful prior and model specification is challenging, especially for Bayesian neural networks. Since deep neural networks are highly nonlinear and overparameterized, the effect of prior specified for their weights on the functions induced is not straightforward (Sun et al., 2019; Ma and Hernández-Lobato, 2021). Second, in practice, approximate Bayesian inference (Jordan et al., 1999; Brooks et al., 2011; Li, 2018) is indispensable since the exact posterior requires

intractable integration. This raises the question about choices and structure of the approximate posterior and an ideal approximation needs to be both accurate and scalable.

This thesis aims to advance Bayesian deep learning specifically for deep sequence models, which now become a unifying modeling tool for widespread applications. Instead of thinking of solutions purely from probabilistic perspective, such as hand-crafting meaningful priors on countless network weights, we propose to build upon the enormous progress of deep sequence models by exploiting the inductive biases within them, such as the network architectures. We leverage these inductive biases to construct probabilistic models and approximation techniques tailored to these models. This not only enables reinforced improvements by combining both deep learning and probabilistic modeling, but also ensures that the methods developed here leave the characteristic mechanisms of deep sequence models intact.

1.1 Thesis Outline and Contributions

The remaining chapters in this thesis are organized as follows.

- Chapter 2 introduces the technical background for the thesis. We review key concepts and ideas in deep sequence models, probabilistic modeling, and Bayesian deep learning, which form the basis of the methods proposed in this thesis. The chapter is concluded with two core applications of deep probabilistic modeling, namely uncertainty quantification and deep generative models.

- Chapter 3 aims at calibrating the uncertainty of Transformer models (Vaswani et al., 2017) by identifying connection between dot-product attention and the posterior mean of sparse Gaussian process (SGP; (Snelson and Ghahramani, 2005)). Based on this key connection, we propose Sparse Gaussian Process attention (SGPA), which replaces the scaled dot-product operation in Transformer with a valid symmetric kernel and uses sparse Gaussian processes techniques to approximate the posterior processes in the output space of multi-head attention blocks (MHAs) directly. Empirically, on a suite of prediction tasks on text, images and graphs, SGPA-based Transformers achieve competitive predictive accuracy, while noticeably improving both in-distribution uncertainty calibration and out-of-distribution robustness and detection. This chapter is based on Chen and Li (2023)

- Chapter 4 proposes a novel online Gaussian process (GP) model (Bui et al., 2017) to capture long-term memory in sequentially observed data in an online learning setting. Our model, Online HiPPO Sparse Variational Gaussian Process

(OHSVGP), leverages the long-range memory modeling capabilities of HiPPO framework (Gu et al., 2020) by interpreting the HiPPO time-varying orthogonal projections as inducing variables capable of memorizing the process history. We show that the HiPPO framework fits naturally into the interdomain GP framework (Van der Wilk et al., 2020) and demonstrate that the kernel matrices can also be updated online in a recurrence form based on the ODE evolution of HiPPO. We evaluate OHSVGP with online prediction for 1D time series, continual learning in discriminative GP model for data with multidimensional inputs, and deep generative modeling with sparse Gaussian process variational autoencoder (Jazbec et al., 2021), showing that it outperforms existing online GP methods in terms of predictive performance, long-term memory preservation, and computational efficiency. This chapter is based on Chen et al. (2025b)

- Chapter 5 is motivated by the superior performance of diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021b) against other sequential generative models, such as hierarchical variational autoencoders (HVAEs; (Sønderby et al., 2016; Maaløe et al., 2019; Vahdat and Kautz, 2020)). We hypothesize that this can be partly attributed to the additional self-supervision information for their intermediate latent states provided by corrupted images, which along with the original image form a pseudo video. Based on this hypothesis, we explore the possibility of improving other types of generative models with such pseudo videos by first extending an image generative model to its video generative model counterpart, and then train the video generative model on pseudo videos constructed by applying data augmentation to the original images. Furthermore, we analyze the potential issues of first-order Markov data augmentation methods, as typically used in diffusion models, in sequential generation from a probabilistic perspective, and propose to use data augmentation with more expressive probabilistic structure to construct more useful information in pseudo videos. We empirically verify the effectiveness of additional self-supervised information from pseudo videos with experiments on the CIFAR10 and CelebA datasets. This chapter is based on Chen et al. (2025a)

- Chapter 6 provides an outlook of our key findings and contributions made in this thesis. Moreover, we identify several open challenges and plausible avenues for future work.

1.2 List of Publications

This section provides a full list of publications that I co-authored during my PhD. Titles are boldfaced for papers whose content is included in this thesis and I also give a brief description of my contribution to each of these works. Some of the material, including ideas, concepts, texts, figures, tables presented in this thesis have previously appeared in these works. The asterisk superscript ($*$) indicates co-first authorship with equal contribution.

Peer-reviewed Conference Publications.

1. (Chen et al., 2025b) Wenlong Chen*, Naoki Kiyohara*, Harrison Bo Hua Zhu*, Jacob Curran-Sebastian, Samir Bhatt, and Yingzhen Li. **Recurrent Memory for Online Interdomain Gaussian Processes**. In *Advances in Neural Processing Systems (NeurIPS)*, 2025.
   My contribution: the main idea was developed by me while the last author provided useful suggestions. The three co-first authors (*) all contributed to the code implementation, experimentation, and paper writing under the supervision of the last author. Moreover, I performed all the derivations and helped orchestrate other authors' contributions.

2. (Jayasekera et al., 2025) I. Shavindra Jayasekera*, Jacob Si*, Filippo Valdettaro, Wenlong Chen, A. Aldo Faisal, and Yingzhen Li. Variational Uncertainty Decomposition for In-Context Learning. In *Advances in Neural Processing Systems (NeurIPS)*, 2025.

3. (Jung et al., 2025a) Yohan Jung, Hyungi Lee, Wenlong Chen, Thomas Möllenhoff, Yingzhen Li, Juho Lee, Mohammad Emtiyaz Khan. Compact Memory for Continual Logistic Regression. In *Advances in Neural Processing Systems (NeurIPS)*, 2025.

4. (Chen et al., 2024) Wenlong Chen*, Yegor Klochkov*, and Yang Liu. Post-hoc Bias Scoring is Optimal for Fair Classification. In *International Conference on Learning Representations (ICLR)*, 2024. Awarded spotlight presentation.

5. (Chen and Li, 2023) Wenlong Chen and Yingzhen Li. **Calibrating Transformers via Sparse Gaussian Processes**. In *International Conference on Learning Representations (ICLR)*, 2023.
   My contribution: the main idea was developed by me while the last author provided useful suggestions. The manuscript was mainly written by me while the last author helped polish it. Moreover, I wrote all the code and performed all the experiments for this work.

Peer-reviewed Workshop Publications.

1. (Chen et al., 2025a) Wenlong Chen*, Wenlin Chen*, Lapo Rastrelli, and Yingzhen Li. **Your Image is Secretly the Last Frame of a Pseudo Video**. In *Deep Generative Model in Machine Learning: Theory, Principle and Efficacy (DeLTa) Workshop at ICLR*, 2025.
   My contribution: the main idea was developed by me while the last author provided useful suggestions. Both co-first authors (∗) contributed to code implementation, experimentation, and paper writing under the supervision of the last author. The third author helped with the experiments related to video diffusion models. I also helped orchestrate other authors' contributions.

2. (Jung et al., 2025b) Yohan Jung*, Hyungi Lee*, Wenlong Chen*, Thomas Möllenhoff, Yingzhen Li, Juho Lee, and Mohammad Emtiyaz Khan. Compact Memory for K-prior Based Continual Learning. In *Symposium on Advances in Approximate Bayesian Inference (AABI)*, 2025.

Preprints.

1. (Chen et al., 2025c) Wenlong Chen*, Bolian Li*, Ruqi Zhang, and Yingzhen Li. Bayesian Computation in Deep Learning. In *arXiv preprint arXiv:2502.18300*, 2025 (to appear as a chapter in Handbook of Markov Chain Monte Carlo - 2nd Edition).

Chapter 2

Background

As two rapidly evolving research areas, both deep sequence models and probabilistic modeling have a vast amount of literature. This chapter aims to provide a compact introduction to relevant concepts and methods in these two fields, based on which the research in this thesis is developed. We begin by reviewing two popular network architectures in deep sequence modeling, namely Transformers (Section 2.1.1), and High-order Polynomial Projection Operators (HiPPO) which establishes the theoretical foundation of deep state space models (SSMs) (Section 2.1.2). Next, we introduce key concepts in probabilistic machine learning and variational inference for approximate posterior (Section 2.2.2). We review in detail a classical class of probabilistic models, Gaussian processes, and common techniques of building deep models with them (Section 2.3). In addition to predictive models, we last review the applications of probabilistic machine learning in the domain of deep generative models, with a particular focus on sequential generative models (Section 2.4).

2.1 Deep Sequence Model Architectures

Sequence modeling is at the core of many machine learning applications nowadays, and the tremendous success of deep learning in this field can be largely attributed to several powerful network architectures. In this section, we review two popular deep sequence model architectures, Transformer and High-order Polynomial Projection Operators (HiPPO).

2.1.1 Transformer

A generic Transformer is constructed by multi-layer perceptron (MLP) and multi-head self-attention (MHSA), and we review these two base architectures below.

Multi-layer Perceptron. Multi-layer perceptron (MLP; (Rosenblatt, 1958)) is an ubiquitous base component in deep neural networks. Given an input $\boldsymbol{x} \in \mathbb{R}^{d_{in}}$, an L-layer MLP $f_{\boldsymbol{\theta}}(\boldsymbol{x}) : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}$ applies a series of transformations to the input as follows:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathbf{W}^L g(\mathbf{W}^{L-1} g(\cdots g(\mathbf{W}^1 \boldsymbol{x} + \boldsymbol{b}^1)) + \boldsymbol{b}^{L-1}) + \boldsymbol{b}^L, \tag{2.1}$$

where we use $\boldsymbol{\theta} = \{\mathbf{W}^l, \boldsymbol{b}^l\}_{l=1}^L$ to denote the collection of learnable weight matrices and bias vectors,

$$\mathbf{W}^1 \in \boldsymbol{R}^{d_h \times d_{in}}, \boldsymbol{b}^1 \in \boldsymbol{R}^{d_{in}}, \mathbf{W}^l \in \boldsymbol{R}^{d_h \times d_h}, \boldsymbol{b}^l \in \boldsymbol{R}^{d_h}, l = 2, ..., L-1,$$
$$\mathbf{W}^L \in \boldsymbol{R}^{d_{out} \times d_h}, \boldsymbol{b}^L \in \boldsymbol{R}^{d_{out}}.$$

Here, the hidden dimension (or width) and the output dimension of this MLP are d_h and d_{out}, respectively. $g(\cdot)$ is an element-wise nonlinear activation, which makes the MLP a non-linear function w.r.t. $\boldsymbol{x}$. Among them, Rectified Linear Unit (ReLU; (Nair and Hinton, 2010)) and its variants (Klambauer et al., 2017) are the most commonly used activation functions. ReLU is a piecewise function that maps positive inputs to itself and negative inputs to 0:

$$\text{ReLU}(s) := \max(0, s). \tag{2.2}$$

Although most modern neural network architectures are more complex than pure MLPs, they are still often used as basic modules within modern architectures, for instance, as expressive neural feature extractors.

Multi-head Attention. Attention mechanism, first introduced in Graves et al. (2013a), has become the characteristic building block for Transformer models. The most widely used attention module in modern Transformers are dot-product based attention (Vaswani et al., 2017; Dosovitskiy et al., 2021). Given a sequence of T_q queries $\boldsymbol{q} \in \mathbb{R}^{T_q \times d_q}$, a sequence of T_k (often $T_q = T_k := T$ as in self-attention discussed below) keys $\boldsymbol{k} \in \mathbb{R}^{T_k \times d_k}$ ($d_k = d_q$), and the values, $\boldsymbol{v} \in \mathbb{R}^{T_k \times d_v}$, associated with the keys, dot-product attention (Vaswani et al., 2017) is computed as follows:

$$\boldsymbol{F} = \omega(\boldsymbol{q}\boldsymbol{k}^\top)\boldsymbol{v}, \tag{2.3}$$

where ω is a nonlinear activation function. The dot-product attention computes the weighted sum of the values, where the weights can be viewed as similarity scores between the queries and keys measured by their dot products, $\boldsymbol{q}\boldsymbol{k}^\top$. For self-attention, the keys are simply set to be equal to the queries, i.e., $\boldsymbol{k} = \boldsymbol{q}$. Transformers employ

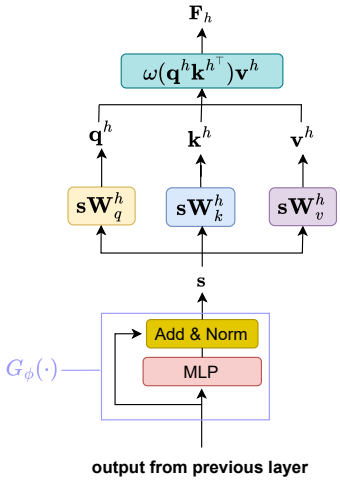


Figure 2.1: Illustration of one head (h) of multi-head self attention in one layer of Transformer.

multi-head self-attention (MHSA) constructed by multiple heads of dot-product attention whose results are aggregated to produce the final output. The queries, keys, and values for each head are all obtained from the same input sequence. Specifically, MHSA modifies dot-product self-attention as follows. Assume H attention heads are in use, then given a sequence of T inputs $\boldsymbol{s} \in \mathbb{R}^{T \times d_s}$ to the MHSA block, we project them to the queries for each head h with a projection matrix $\boldsymbol{W}_q^h \in \mathbb{R}^{d_s \times d_q}$: $\boldsymbol{q}^h = \boldsymbol{s}\boldsymbol{W}_q^h$. We obtain the keys $\boldsymbol{k}^h$ and values $\boldsymbol{v}^h$ accordingly by projections using matrices $\boldsymbol{W}_k^h \in \mathbb{R}^{d_s \times d_k}$ and $\boldsymbol{W}_v^h \in \mathbb{R}^{d_s \times d_v}$ respectively. Typically, we use the same $d_q = d_k = d_v$ for all the heads. Then the head's output $\boldsymbol{F}_h$ is obtained by plugging $\boldsymbol{q}^h$, $\boldsymbol{k}^h$ and $\boldsymbol{v}^h$ to Eq. 2.3. Lastly, the attention outputs from each head is combined with the output projection matrix $\boldsymbol{W}_F \in \mathbb{R}^{(Hd_v) \times (Hd_v)}$ as follows:

$$\boldsymbol{F} = \text{concat}(\boldsymbol{F}_1, \cdots, \boldsymbol{F}_H)\boldsymbol{W}_F. \tag{2.4}$$

In Transformers, multiple layers of MHSA may be stacked together, where the output of the $(l-1)$-th MHSA layer is further processed by a nonlinear function $G_{\boldsymbol{\phi}^l}$ – parameterized by an MLP with common deep learning tricks (e.g., residual connection (He et al., 2015) and layer normalization (Ba et al., 2016)) – to obtain the input to the l-th MHSA layer, i.e., $\boldsymbol{s}^l = G_{\boldsymbol{\phi}^l}(\boldsymbol{F}^{l-1})$. Figure 2.1 illustrates an MHSA block in a Transformer model (excluding the combination projection step of Eq. 2.4).

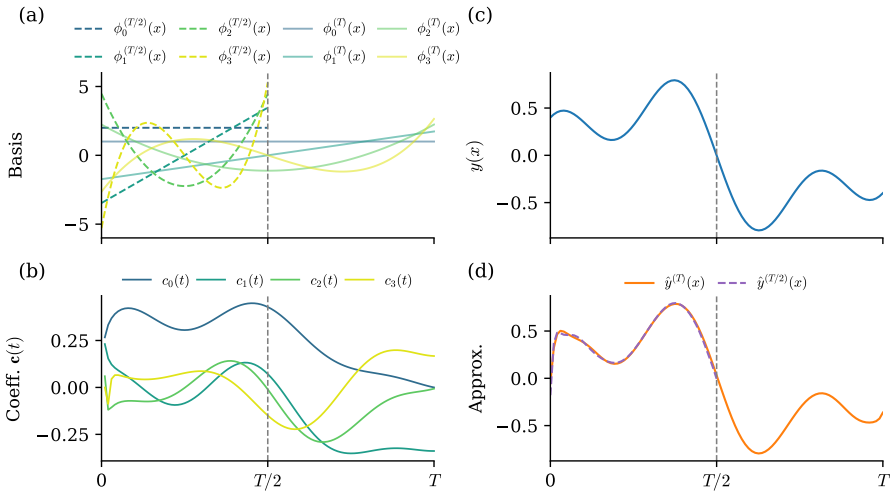


Figure 2.2: An illustration of HiPPO-LegS approximation of a toy time series (c) $y(x)$. Here x is used to denote arbitrary time index. (a) Time-dependent basis functions with end time index $t = T/2$ and $t = T$ ($T = 1$ here). (b) Evolution of the memory state $\boldsymbol{c}(t)$. (d) Finite basis approximation $\hat{y}^t(x)$ with the end time index $t = T/2$ and T.

2.1.2 HiPPO: Recurrent Memory with Optimal Polynomial Projections

The HiPPO framework (Gu et al., 2020) is proposed to build a compact representation that captures long-term dependencies in sequences, which is a major challenge in long sequence modeling. HiPPO provides mathematical foundations for compressing continuous-time signals into finite-dimensional memory states through optimal polynomial projections. Given a time series $y(t)$, HiPPO maintains a memory state $\boldsymbol{c}(t) \in \mathbb{R}^M$ that optimally approximates the historical signal $\{y(x)\}_{x \le t}$. The framework consists of a time-dependent measure $\omega^{(t)}(x)$ over $(-\infty, t]$ that defines input importance, along with adaptive polynomial basis functions $\{g_m^{(t)}(x)\}_{m=0}^{M-1}$ obtained from adapting the domain of the basis functions to cover the new time region. For example, the standard Legendre polynomials $P_m(x)$ considered in HiPPO-LegS variant have domain $[-1, 1]$, and when the last historical time instance is t (assuming to be positive), we define the adaptive basis

$$g_m^{(t)}(x) := (2m+1)^{1/2} P_m\left(\frac{2x}{t} - 1\right), \tag{2.5}$$

which has the domain $[0, t]$. Moreover, these adaptive bases are orthonormal under certain $\omega^{(t)}(x)$ (e.g., for adaptive Legendre polynomials above, the corresponding $\omega^{(t)}(x)$ is uniform over $[0, t]$):

$$\int_{-\infty}^{t} g_m^{(t)}(x) g_l^{(t)}(x) \omega^{(t)}(x) \mathrm{d}x = \delta_{ml}. \tag{2.6}$$

The historical signal is encoded through projection coefficients given by

$$c_m(t) = \int_{-\infty}^{t} y(x) g_m^{(t)}(x) \omega^{(t)}(x) \mathrm{d}x, \tag{2.7}$$

which yields the approximation

$$y(x) \approx \hat{y}^{(t)}(x) = \sum_{m=0}^{M-1} c_m(t) g_m^{(t)}(x), \tag{2.8}$$

for $x \in (-\infty, t]$. The compression is usually lossy unless $y(x)$ lives in the span of the finite bases. However, $\{c_m(t)\}_{m=0}^{M-1}$ obtained in Eq. 2.7 is optimal in the sense that it minimizes the following L^2-error:

$$\{c_n\}_{n=0}^{M-1} = \underset{\{c'_n\}_{n=0}^{M-1}}{\arg\min} \int_{-\infty}^{t} \|y(x) - \sum_{n=0}^{M-1} c'_n(t) g_n(x)\|^2 \omega^{(t)}(x) \mathrm{d}x. \tag{2.9}$$

Hence, the M-dimensional memory state $\boldsymbol{c}(t) := [c_0(t), \ldots, c_{M-1}(t)]^\top$ forms a compact representation that optimally captures the historical information in $y(x)$ up to $x = t$.

This memory state $\boldsymbol{c}(t)$ is a function of the end point of the history t. Every time when observation at new time instance, $t + \Delta t$ appear, we adapt the basis to $t + \Delta t$, and compute $\boldsymbol{c}(t + \Delta t)$. A neat property of the memory state is that it enables online updates: we do not need to recompute the integrals from scratch to obtain the memory state each time we observe new data, instead we can obtain $\boldsymbol{c}(t + \Delta t)$ using an online update based on $\boldsymbol{c}(t)$. The time derivative of the adaptive polynomial basis functions considered in HiPPO, $\frac{\partial}{\partial t} g_m^{(t)}(x)$, can be expressed as a linear combination of lower-order polynomial bases $\{g_l^{(t)}(x)\}_{l=0}^{m-1}$. Consequently, the time derivative of $\boldsymbol{c}(t)$ w.r.t. t induces a linear ordinary differential equation (ODE)

$$\frac{\mathrm{d}}{\mathrm{d}t} \boldsymbol{c}(t) = \boldsymbol{A}(t) \boldsymbol{c}(t) + \boldsymbol{B}(t) y(t), . \tag{2.10}$$

Discretizing the linear ODE yields the recurrence of the form

$$\boldsymbol{c}_k = \boldsymbol{A}_k \boldsymbol{c}_{k-1} + \boldsymbol{B}_k y_k, \tag{2.11}$$

which allows online updates without having to revisit the past signals. Here, $\boldsymbol{A}_k$ and $\boldsymbol{B}_k$ are determined by $\boldsymbol{A}(t)$, $\boldsymbol{B}(t)$ and the discretization step size.

HiPPO supports various measure-basis configurations which can enable online updates of the corresponding memory state (Gu et al., 2020, 2023). Here, we provide some details for a canonical instantiation, HiPPO-LegS. For a given end time index t,

HiPPO-LegS uses the uniform measure $\omega^{(t)}(x) = \frac{1}{t}\mathbf{1}_{[0,t]}(x)$ and scaled Legendre polynomials with input domain adapted to $[0,t]$, $g_m^{(t)}(x) = (2m+1)^{1/2}P_m\left(\frac{2x}{t} - 1\right)$, as basis functions, where $P_m(\cdot)$ is the canonical m-th Legendre polynomial with input domain $[-1,1]$, and $\mathbf{1}_{[0,t]}(x)$ is the indicator function on the interval $[0,t]$. The uniform measure over the past provides the inductive bias that encourages HiPPO-LegS to keep the whole past in memory. The memory state of HiPPO-LegS evolves according to a linear ODE as in Eq. 2.10. To see this, we define $\phi_m^{(t)}(x) := g_m^{(t)}(x)\omega^{(t)}(x)$. The time derivative of $\phi_m^{(t)}(x)$ can be shown to be:

$$\begin{aligned}\frac{\partial}{\partial t}\phi_m^{(t)}(x) &= \omega_m^{(t)}(x)\frac{\partial}{\partial t}g_m^{(t)}(x) + g_m^{(t)}(x)\frac{\partial}{\partial t}\omega_m^{(t)}(x) \\ &= -\frac{\sqrt{2m+1}}{t}\Big[\frac{m+1}{\sqrt{2m+1}}\phi_m^{(t)}(x) + \sqrt{2m-1}\phi_{m-1}^{(t)}(x) \\ &\qquad + \sqrt{2m-3}\phi_{m-2}^{(t)}(x) + \cdots\Big] + \frac{1}{t}\,\delta_t(x),\end{aligned} \tag{2.12}$$

where $\delta_t(x)$ is the Dirac delta at $x = t$ and it arises from the time derivative of the uniform measure $\omega^{(t)}(x)$ (see Appendix D.3 in Gu et al. (2020) for details). Plugging this expression into

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{c}(t) &= \Big[\frac{\mathrm{d}}{\mathrm{d}t}c_0(t), \cdots, \frac{\mathrm{d}}{\mathrm{d}t}c_{M-1}(t)\Big]^\top \\ &= \Big[\int y(x)\frac{\partial}{\partial t}\phi_0^{(t)}(x)dx, \cdots, \int y(x)\frac{\partial}{\partial t}\phi_{M-1}^{(t)}(x)dx\Big]^\top\end{aligned} \tag{2.13}$$

allows us to identify the matrices $\boldsymbol{A}(t) \in \mathbb{R}^{M\times M}$ and $\boldsymbol{B}(t) \in \mathbb{R}^{M\times 1}$ in the ODE update of the memory state (Eq. 2.10). For HiPPO-LegS, their explicit formulas are given by:

$$\boldsymbol{A}(t) = \frac{1}{t}\boldsymbol{A}, \quad [\boldsymbol{A}]_{mk} = \begin{cases} -\sqrt{(2m+1)(2k+1)} & \text{if } m > k \\ -m-1 & \text{if } m = k \\ 0 & \text{if } m < k \end{cases} \tag{2.14}$$

and

$$[\boldsymbol{B}(t)]_m = \frac{[\boldsymbol{B}]_m}{t} = \frac{\sqrt{2m+1}}{t}, \tag{2.15}$$

where the factor $1/t$ reflects the time-dependent scaling of the basis functions to the adaptive interval $[0,t]$. Figure 2.2 shows evolution of the memory state for a toy time series in HiPPO-LegS, and the approximation of the time series based on the memory state and adaptive Legendre polynomials for two different end time indices of the history.

We include details of other HiPPO variants which use different combinations of

function basis and measures in Appendix A.1. Compared with HiPPO-LegS, they use measures allocating more importance over the more recent history so that they are less suitable for tasks requiring long-term memory in sequences.

A few recent works extend the HiPPO recurrences for more efficient long-range memory modeling and stack them along with MLPs to build deep SSMs, which is now a class of architectures competitive to Transformers. The structured state space sequential (S4) model (Gu et al., 2022) extends HiPPO with trainable recurrence parameters and accelerates the computation with convolutional kernels, while Mamba (Gu and Dao, 2023; Dao and Gu, 2024) further introduces hardware-aware selective state mechanisms.

2.2 Scalable Probabilistic Learning

Probabilistic machine learning, based on rules of probability and Bayesian statistics, offers a principled framework for systematically expressing and updating our knowledge about unknowns under uncertainty. In particular, Bayesian inference leverages Bayes' rule (Bayes, 1763) to update our belief about the unobserved variables after observing the data.

2.2.1 Bayesian Inference

Given a machine learning model specified by some parameters $\boldsymbol{\theta}$ (e.g., weights of a deep neural network), we turn the model into a probabilistic model by treating the parameters as unobserved random variables and placing a prior distribution, $p(\boldsymbol{\theta})$, over them, which represents our initial belief about the model parameters before observing any data. For instance, it can be used to rule out extreme parameter values that can lead to erratic predictions. The data $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{n=1}^N$ are treated as observed variables and the information for $\boldsymbol{\theta}$ from the data, also known as likelihood $p(\mathcal{D}|\boldsymbol{\theta})$, is used to refine the prior into the posterior:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\boldsymbol{\theta}, \mathcal{D})}{p(\mathcal{D})} = \frac{p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta})}{\int p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta})d\boldsymbol{\theta}}. \tag{2.16}$$

The model uncertainty in $\boldsymbol{\theta}$ from the posterior is then translated to predictive uncertainty associated with the prediction at new input $\boldsymbol{x}^*$, via the posterior predictive distribution defined by the following marginalization:

$$p(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) = \int p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}. \tag{2.17}$$

In addition to the uncertainty quantification capability, Bayesian inference is naturally suitable for online learning (Cesa-Bianchi and Lugosi, 2006; Ritter et al., 2018) or continual learning (Kirkpatrick et al., 2017; Nguyen et al., 2018), where data points are observed sequentially in batches. When new data arrive, the previous posterior obtained based on old data, $p(\boldsymbol{\theta}|\mathcal{D}_{old})$, becomes the new prior and we combine the information from the current data, $\mathcal{D}_{new}$, to form the new posterior again via Bayes' rule:

$$p(\boldsymbol{\theta}|\mathcal{D}_{new}, \mathcal{D}_{old}) = \frac{p(\boldsymbol{\theta}|\mathcal{D}_{old})p(\mathcal{D}_{new}|\boldsymbol{\theta}, \mathcal{D}_{old})}{\int p(\boldsymbol{\theta}|\mathcal{D}_{old})p(\mathcal{D}_{new}|\boldsymbol{\theta}, \mathcal{D}_{old})d\boldsymbol{\theta}}, \tag{2.18}$$

This procedure can be repeated whenever new data arrive, enabling online or continual learning under uncertainty.

Unfortunately, exact Bayesian inference as in Eq. 2.16 is typically intractable since it requires solving a complex integral for the marginal likelihood in the denominator, $p(\mathcal{D}) = \int p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta})d\boldsymbol{\theta}$, which has no analytic solution unless the prior is conjugate to the likelihood. Hence, in practice, approximate inference techniques (Li, 2018), which approximate the target posterior with an approximate posterior $q(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}|\mathcal{D})$, become indispensable. In such case, the posterior predictive distribution in Eq. 2.17 can also be approximated based on $q(\boldsymbol{\theta})$ as follows:

$$p(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) \approx \int p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta})q(\boldsymbol{\theta})d\boldsymbol{\theta} \approx \frac{1}{M}\sum_{m=1}^{M} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}_m), \quad \boldsymbol{\theta}_m \sim q(\boldsymbol{\theta}), \tag{2.19}$$

where a further Monte Carlo estimation is typically applied for marginalization.

In this thesis, we use variational inference (Jordan et al., 1999; Beal, 2003; Li, 2018; Zhang et al., 2018a), an optimization-based approximate inference technique, which is much more scalable than simulation-based techniques, such as Markov chain Monte Carlo (Metropolis et al., 1953; Gelfand, 2000; Ma et al., 2015). This makes variational inference a popular method for approximate inference in deep learning applications, where scalability is crucial.

2.2.2 Variational Inference

Variational inference (VI) (Beal, 2003; Jordan et al., 1999) turns the inference problem into an optimization procedure. It first specifies a parametric distribution family, $\mathcal{Q} := \{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})\}$, with tunable parameters $\boldsymbol{\phi}$, such as factorized Gaussians where $\boldsymbol{\phi}$ are the mean and variance parameters. Then we optimize over $\mathcal{Q}$ to find the best approximate posterior (also called optimal variational distribution), $q_{\boldsymbol{\phi}^*}(\boldsymbol{\theta})$, which is defined as the one closest to the target posterior $p(\boldsymbol{\theta}|\mathcal{D})$ within the parametric family, measured by the Kullback-Leibler (KL) divergence. Specifically, the procedure requires the minimization of KL-divergence w.r.t. the parameters $\boldsymbol{\phi}$ of the chosen

parametric distribution family to obtain the optimal variational distribution:

$$\begin{aligned}\boldsymbol{\phi}^* &= \arg\min_{q_{\boldsymbol{\phi}} \in \mathcal{Q}} \mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \| p(\boldsymbol{\theta}|\mathcal{D})), \\ \mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \| p(\boldsymbol{\theta}|\mathcal{D})) &= \int q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\mathcal{D})} d\boldsymbol{\theta}.\end{aligned} \tag{2.20}$$

We can then substitute in $q_{\boldsymbol{\phi}^*}(\boldsymbol{\theta})$ for $q(\boldsymbol{\theta})$ in Eq. 2.19 for approximate posterior predictive distribution.

However, the KL-divergence requires the evaluation of $p(\boldsymbol{\theta}|\mathcal{D})$, and its dependence on the intractable marginal likelihood (also known as model evidence) $p(\mathcal{D})$ is the motivation for approximate inference in the first place. Therefore, direct minimization of KL-divergence is infeasible, and an alternative objective is needed for its minimization, which prompts the following evidence lower bound (ELBO) as objective. The ELBO is a lower bound of the log marginal likelihood and it is obtained by subtracting the KL-divergence from the log marginal likelihood, which is a constant w.r.t. $\boldsymbol{\phi}$. Hence, minimization of the KL-divergence can be achieved indirectly via the maximization of the (ELBO).

$$\begin{aligned}\mathcal{L}_{ELBO}(\boldsymbol{\phi}) &:= \log p(\mathcal{D}) - \mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \| p(\boldsymbol{\theta}|\mathcal{D})) \\ &= \log p(\mathcal{D}) - \int q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) p(\mathcal{D})}{p(\boldsymbol{\theta}) p(\mathcal{D}|\boldsymbol{\theta})} d\boldsymbol{\theta} \\ &= \int q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \log \frac{p(\boldsymbol{\theta}) p(\mathcal{D}|\boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})} d\boldsymbol{\theta} \\ &= \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}[\log p(\mathcal{D}|\boldsymbol{\theta})]}_{\text{ELL}} - \underbrace{\mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \| p(\boldsymbol{\theta}))}_{\text{KL regularizer}}\end{aligned} \tag{2.21}$$

The above ELBO is decomposed into two terms. The first term is commonly called the expectation of log-likelihood (ELL), and the second term is a KL regularizer between the approximate posterior $q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$ and prior $p(\boldsymbol{\theta})$. The ELL term encourages $q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$ to place more importance over the model configurations that explain the data, while the second term regularizes $q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$ toward the prior. The combination of the two terms balance the data fit and the regularization.

In practice, unless with special conditions (e.g., Gaussianity) for the specification of the prior, likelihood, and the variational distribution, we may not be able to analytically compute one or both terms in ELBO. Hence, typically a Monte Carlo estimation is further employed:

$$\mathcal{L}_{ELBO} = \frac{1}{M} \sum_{m=1}^{M} \left[\log p(\mathcal{D}|\boldsymbol{\theta}_m) - \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}_m) + \log p(\boldsymbol{\theta}_m)\right], \qquad \boldsymbol{\theta}_m \overset{\text{iid}}{\sim} q_{\boldsymbol{\phi}}(\boldsymbol{\theta}). \tag{2.22}$$

If the chosen variational distribution family, $\mathcal{Q}$, contains the true posterior, VI will return it, if the optimization is solved perfectly. In practice, due to the computational budget and tractability of distribution families, a simple yet tractable $\mathcal{Q}$ is often considered to enable fast computation and reduced memory complexity.

To exploit gradient based optimization of the ELBO, the *reparameterization trick* (Kingma and Welling, 2014) is proposed for a variety of variational distribution families including Gaussians. This approach makes the sampling operation $\boldsymbol{\theta} \sim q_{\boldsymbol{\phi}}(\boldsymbol{\theta})$ differentiable w.r.t. $\boldsymbol{\phi}$ by viewing it as passing an auxiliary noise variable $\boldsymbol{\epsilon} \sim p_{base}(\boldsymbol{\epsilon})$ through a function $T_{\boldsymbol{\phi}}(\boldsymbol{\epsilon})$ that is differentiable w.r.t. $\boldsymbol{\phi}$:

$$\boldsymbol{\theta} \sim q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \quad \Leftrightarrow \quad \boldsymbol{\theta} = T_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}),\ \boldsymbol{\epsilon} \sim p_{base}(\boldsymbol{\epsilon}). \tag{2.23}$$

For instance, for D-dimensional factorized Gaussians with mean $\boldsymbol{\mu} = [\mu_1, \cdots, \mu_d]^\top$ and variance $\boldsymbol{\sigma}^2 = [\sigma_1^2, \cdots, \sigma_d^2]^\top$, $q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) = \prod_{d=1}^{D} \mathcal{N}(\boldsymbol{\theta}_d; \mu_d, \sigma_d)$, this sampling operation can be written as $\boldsymbol{\theta} = T_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}) := \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$, with $\boldsymbol{\epsilon}_d \sim \mathcal{N}(0, 1)$, and $\odot$ denotes element-wise multiplication. The Monte Carlo estimate of the ELL term in Eq. 2.21 can therefore be rewritten using the change-of-variable rule as follows:

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}[\log p(\mathcal{D}|\boldsymbol{\theta})] \approx \frac{1}{M} \sum_{m=1}^{M} \log p(\mathcal{D}|\boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}_m), \quad \boldsymbol{\epsilon}_m \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \boldsymbol{I}). \tag{2.24}$$

Moreover, with likelihood that factorizes w.r.t. the data points, stochastic optimization methods based on mini-batch estimation of the above ELL term, such as stochastic gradient descent (SGD), can be used to scale VI to large datasets.

VI has been used extensively for scalable inference in probabilistic models, such as Gaussian processes and Bayesian neural networks, and we will review its applications in Gaussian process models in detail next. I would be remiss to not review some excellent works about weight space VI in the context of Bayesian neural networks since they are not directly related to the research in this thesis. We refer the interested readers to Arbel et al. (2023) for an overview of the topic.

2.3 Gaussian Processes

Gaussian processes (GPs; (Rasmussen and Williams, 2006)) is a class of powerful Bayesian non-parametric models that are widely used to infer unknown functions under uncertainty. Instead of assuming a parametric form of the model and performing posterior inference over parameters, GPs directly specify distributions over function values. Informally, GPs can be viewed as infinite-dimensional Gaussian distributions for the function values, evaluated over an index set $\mathcal{X}$ (domain of f), which can be

infinite, e.g., $\mathcal{X} = \mathbb{R}^D$.

A GP is fully specified by a mean function $m_{\boldsymbol{\varphi}}(\cdot) : \mathcal{X} \to \mathbb{R}$ and a symmetric positive-definite covariance covariance function, typically parameterized by a kernel function $k_{\boldsymbol{\varphi}}(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, $p(f; \boldsymbol{\varphi}) = \mathcal{GP}(f; m_{\boldsymbol{\varphi}}(\cdot), k_{\boldsymbol{\varphi}}(\cdot, \cdot))$. Here, $\boldsymbol{\varphi}$ is used to to denote the hyperparameters associated with the mean function and the kernel function.

GP provides a powerful Bayesian modeling paradigm in the function space by first placing a GP prior over the unknown function f. Typically, an uninformative prior mean function is in use, e.g., zero function $m(\cdot) := 0$, and this convention will be followed throughout this thesis.

$$p(f; \boldsymbol{\varphi}) = \mathcal{GP}(f; 0, k_{\boldsymbol{\varphi}}(\cdot, \cdot)). \tag{2.25}$$

Notably, as stochastic processes, a core property of GPs is marginal consistency. For example, the marginal distribution according to the above GP prior for function values $\boldsymbol{f_X} := [f(\boldsymbol{x}_1), \cdots, f(\boldsymbol{x}_N)]^\top$ over a finite subset of input locations, $\boldsymbol{X} := [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]^\top \in \mathcal{X}$, is a multivariate Gaussian as follows:

$$p(\boldsymbol{f_X}; \boldsymbol{\varphi}) = \mathcal{N}(\boldsymbol{f_X}; \boldsymbol{0}, \boldsymbol{K_{XX}}), \tag{2.26}$$

where the covariance matrix is obtained via the kernel matrix, $\boldsymbol{K_{XX}}$, whose ij-th element is the kernel value between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, $[\boldsymbol{K_{XX}}]_{ij} = k_{\boldsymbol{\varphi}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Notice that $\boldsymbol{K_{XX}}$ depends on hyperparameters $\boldsymbol{\varphi}$, but we do not include $\boldsymbol{\varphi}$ in its notation for consice presentation.

2.3.1 Exact Inference for Regression

In regression problems with observed data $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{y}) := \{\boldsymbol{x}_n, y_n\}_{n=1}^N$, typically an i.i.d Gaussian likelihood with noise variance σ^2 is assumed,

$$p(\boldsymbol{y}|\boldsymbol{f_X}; \sigma^2) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{f_X}, \sigma^2 \boldsymbol{I}), \tag{2.27}$$

the posterior process is also a GP since the Gaussian prior and likelihood are conjugate, and the following closed-form posterior predictive distribution can be obtained:

$$\begin{aligned} p(\boldsymbol{y}^*|\boldsymbol{X}^*, \mathcal{D}; \boldsymbol{\varphi}) = \mathcal{N}(\boldsymbol{y}^*; \boldsymbol{K_{X^*X}}(\boldsymbol{K_{XX}} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{y}, \\ \boldsymbol{K_{X^*X^*}} + \sigma^2 \boldsymbol{I} \underbrace{- \boldsymbol{K_{X^*X}}(\boldsymbol{K_{XX}} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{K_{XX^*}}}_{\text{uncertainty reduction}}), \end{aligned} \tag{2.28}$$

where $\boldsymbol{X}^*$ are test inputs. The last term in the posterior predictive covariance is commonly referred to as uncertainty reduction, and its magnitude is determined by the similarity between test inputs $\boldsymbol{X}^*$ and observed inputs $\boldsymbol{X}$, measured by the kernel. For $\boldsymbol{X}^*$ closer to $\boldsymbol{X}$, $\boldsymbol{K}_{\boldsymbol{X}^*\boldsymbol{X}}$ will be large, resulting in a larger uncertainty reduction. For instance, if kernels based on Euclidean distance, such as RBF kernel, are used, the predictive uncertainty will be Euclidean distance-aware. In general, the kernel can be specified to encode some prior assumptions for the underlying function (e.g., smoothness, periodicity).

The kernel hyperparameters $\boldsymbol{\varphi}$ (e.g., the length-scale in RBF kernel) along with the observation noise variance σ^2 can influence the predictions of GPs noticeably, and a common practice to select them is via the maximization of log marginal likelihood (also known as type-II maximum likelihood estimation). For regression problems, the log marginal likelihood also admits a closed form as follows:

$$\begin{aligned}\log p(\boldsymbol{y}|\boldsymbol{X};\boldsymbol{\varphi}) &= \log \int p(\boldsymbol{y}|\boldsymbol{f}_{\boldsymbol{X}};\boldsymbol{\varphi})p(\boldsymbol{f}_{\boldsymbol{X}};\boldsymbol{\varphi})d\boldsymbol{f}_{\boldsymbol{X}} \\ &= \log \mathcal{N}(\boldsymbol{y};\boldsymbol{0},\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}+\sigma^2\boldsymbol{I}) \\ &= \underbrace{-\frac{1}{2}\boldsymbol{y}^\top(\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}+\sigma^2\boldsymbol{I})^{-1}\boldsymbol{y}}_{\text{data fit}} \underbrace{-\frac{1}{2}\log\det(\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}+\sigma^2\boldsymbol{I})}_{\text{model complexity}} + \text{const.}\end{aligned} \tag{2.29}$$

It consists of two terms that balance between the data fit and the model complexity penalty, which ultimately selects the configuration of hyperparameters corresponding to the simplest model that explains the data well.

Interestingly, the posterior inference of GP can be reformulated through the lens of Bayesian linear regression with kernel feature map (Rasmussen and Williams, 2006, Chapter 2). Consider the linear regression problem as follows.

$$y(x) = \boldsymbol{w}^\top\Phi(\boldsymbol{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0,\sigma^2), \tag{2.30}$$

where $\Phi(\cdot) : \mathcal{X} \to \mathbb{R}^{d_\Phi}$ (d_Φ can be ∞) is the feature map associated with a kernel $k(\cdot,\cdot)$ such that $k(\boldsymbol{x},\boldsymbol{x}') = \Phi(\boldsymbol{x})^\top\Phi(\boldsymbol{x}')$. Mercer's Theorem (Mercer, 1909) guarantees the existence of such feature maps for kernel functions meeting Mercer's condition. A standard Gaussian prior over the weight, $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{0},\boldsymbol{I})$, induces a GP prior in the function space: given data $\mathcal{D} = (\boldsymbol{X},\boldsymbol{y})$, we know that the distribution of function values evaluated over $\boldsymbol{X}$ is a Gaussian due to the linearity of the model, and its mean and covariance can be shown to be $\boldsymbol{0}$ and $\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}+\sigma^2\boldsymbol{I}$, respectively, which exactly

matches the prior for $\boldsymbol{y}_{\boldsymbol{X}}$ in GP model. The posterior of weight is a Gaussian

$$\begin{aligned} p(\boldsymbol{w}|\mathcal{D}) &= \mathcal{N}(\boldsymbol{w}; \boldsymbol{m}_{\boldsymbol{w}}, \boldsymbol{S}_{\boldsymbol{w}}), \\ \boldsymbol{m}_{\boldsymbol{w}} &= \frac{1}{\sigma^2}\boldsymbol{S}_{\boldsymbol{w}}\boldsymbol{\Phi}_{\boldsymbol{X}}^\top \boldsymbol{y}, \\ \boldsymbol{S}_{\boldsymbol{w}} &= (\boldsymbol{I} + \frac{1}{\sigma^2}\boldsymbol{\Phi}_{\boldsymbol{X}}^\top \boldsymbol{\Phi}_{\boldsymbol{X}})^{-1} \\ &= \boldsymbol{I} - \boldsymbol{\Phi}_{\boldsymbol{X}}^\top(\sigma^2\boldsymbol{I} + \boldsymbol{\Phi}_{\boldsymbol{X}}\boldsymbol{\Phi}_{\boldsymbol{X}}^\top)^{-1}\boldsymbol{\Phi}_{\boldsymbol{X}} \qquad \text{(Woodbury identity)} \\ &= \boldsymbol{I} - \boldsymbol{\Phi}_{\boldsymbol{X}}^\top(\sigma^2\boldsymbol{I} + \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}})^{-1}\boldsymbol{\Phi}_{\boldsymbol{X}}, \end{aligned} \tag{2.31}$$

where $\boldsymbol{\Phi}_{\boldsymbol{X}} \in \mathbb{R}^{N \times d_\phi}$ is the design matrix, and notice that $\boldsymbol{\Phi}_{\boldsymbol{X}}\boldsymbol{\Phi}_{\boldsymbol{X}}^\top = \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}$. The posterior predictive for new input locations $\boldsymbol{X}^*$ can be derived based on the weight posterior and it is a Gaussian as follows:

$$\begin{aligned} p(\boldsymbol{y}^*|\boldsymbol{X}^*, \mathcal{D}) &= \int p(\boldsymbol{y}^*|\boldsymbol{w}, \boldsymbol{X}^*)p(\boldsymbol{w}|\mathcal{D})d\boldsymbol{w} \\ &= \mathcal{N}(\boldsymbol{y}^*; \boldsymbol{\Phi}_{\boldsymbol{X}^*}\boldsymbol{m}_{\boldsymbol{w}}, \sigma^2\boldsymbol{I} + \boldsymbol{\Phi}_{\boldsymbol{w}}^\top \boldsymbol{S}_{\boldsymbol{w}}\boldsymbol{\Phi}_{\boldsymbol{w}}). \end{aligned} \tag{2.32}$$

It is easy to see that the posterior predictive covariance is exactly the same as the posterior GP in Eq. 2.28. The posterior predictive mean can be further simplified below:

$$\begin{aligned} \boldsymbol{\Phi}_{\boldsymbol{X}^*}\boldsymbol{m}_{\boldsymbol{w}} &= \frac{1}{\sigma^2}\boldsymbol{\Phi}_{\boldsymbol{X}^*}\boldsymbol{S}_{\boldsymbol{w}}\boldsymbol{\Phi}_{\boldsymbol{X}}^\top \boldsymbol{y} \\ &= \boldsymbol{K}_{\boldsymbol{X}^*\boldsymbol{X}}\left\{\frac{1}{\sigma^2}\left[\boldsymbol{I} - (\sigma^2\boldsymbol{I} + \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}})^{-1}\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}\right]\right\}\boldsymbol{y} \\ &= \boldsymbol{K}_{\boldsymbol{X}^*\boldsymbol{X}}(\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y}. \end{aligned} \tag{2.33}$$

For the last line, simply notice that

$$(\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma^2\boldsymbol{I})\left\{\frac{1}{\sigma^2}\left[\boldsymbol{I} - (\sigma^2\boldsymbol{I} + \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}})^{-1}\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}\right]\right\} = \frac{1}{\sigma^2}\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \boldsymbol{I} - \frac{1}{\sigma^2}\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} = \boldsymbol{I}.$$

Hence, a GP model is equivalent to a Bayesian linear regression model based on kernel feature map.

2.3.2 Sparse Variational Gaussian Process

Although GP exhibits exceptional performance and provides reliable predictive uncertainty in the small data regime, its posterior inference requires matrix inversion with a cubic computational cost w.r.t. the number of observed data points ($O(N^3)$), which makes it computationally intractable for large dataset. One way to extend GP to big data applications is working with an approximate posterior GP with low rank structure in the framework of VI. One popular such approach is sparse

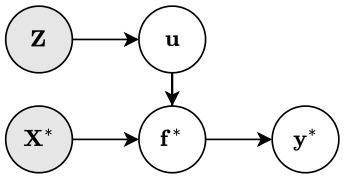


Figure 2.3: The graphical model of prediction of sparse variational Gaussian process (SVGP).

variational Gaussian process (SVGP; (Snelson and Ghahramani, 2005)), which approximates the posterior process with a GP, Q, constructed based on a small set of augmented inducing points (pseudo input-function value pairs) of size M (typically $M \ll N$), $\{(\boldsymbol{z}_m, u_m)\}_{m=1}^{M}$ (Hensman et al., 2013, 2015). SVGP reduces the $O(N^3)$ computational cost of full GP to $O(M^3 + NM^2)$, since, by construction, Q is fully determined by the approximate posterior of the small set of inducing points, which is set to be a Gaussian with learnable variational mean, $\boldsymbol{m_u}$, and variational covariance, $\boldsymbol{S_u}$, $q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{m_u}, \boldsymbol{S_u})$.

According to the standard GP prior, $p(f) = \mathcal{GP}(0(\cdot), k_{\boldsymbol{\varphi}}(\cdot,\cdot))$, the prior marginal distribution of the augmented set consisting of $\boldsymbol{u}$ and $\boldsymbol{f}^*$, evaluated at any arbitrary set of input locations $\boldsymbol{X}^*$, is

$$p(\boldsymbol{f}^*, \boldsymbol{u}|\boldsymbol{X}^*, \boldsymbol{Z}) = \mathcal{N}(\boldsymbol{0}, \begin{pmatrix} \boldsymbol{K_{X^*X^*}} & \boldsymbol{K_{X^*Z}} \\ \boldsymbol{K_{ZX^*}} & \boldsymbol{K_{ZZ}}. \end{pmatrix}). \tag{2.34}$$

The aforementioned computational gain is achieved by imposing the following low rank structure for the approximate posterior GP. Given the approximate posterior of the inducing points, $q(\boldsymbol{u})$, the approximate posterior conditional distribution of $\boldsymbol{f}^*$ given $\boldsymbol{u}$ is assumed to be the same as the prior conditional distribution.

$$q(\boldsymbol{f}^*|\boldsymbol{u}, \boldsymbol{Z}, \boldsymbol{X^*}) = p(\boldsymbol{f}^*|\boldsymbol{u}, \boldsymbol{Z}, \boldsymbol{X^*}). \tag{2.35}$$

Hence, the induced approximate posterior for $\boldsymbol{f}^*$ is a Gaussian with a low rank structure defined by the inducing points:

$$\begin{aligned} q(\boldsymbol{f}^*|\boldsymbol{X}^*, \boldsymbol{Z}) &= \int p(\boldsymbol{f}^*|\boldsymbol{u}, \boldsymbol{Z}, \boldsymbol{X^*}) q(\boldsymbol{u}) d\boldsymbol{u} \\ &= \mathcal{N}(\boldsymbol{f}^*; \boldsymbol{K_{X^*Z}} \boldsymbol{K_{ZZ}^{-1}} \boldsymbol{m_u}, \boldsymbol{K_{X^*X^*}} + \boldsymbol{K_{X^*Z}} \boldsymbol{K_{ZZ}^{-1}} (\boldsymbol{S_u} - \boldsymbol{K_{ZZ}}) \boldsymbol{K_{ZZ}^{-1}} \boldsymbol{K_{ZX^*}}). \end{aligned} \tag{2.36}$$

Note that now the matrix inversion is applied to $\boldsymbol{K_{ZZ}}$, with a $O(M^3)$ computational cost, which is significantly lower than the $O(N^3)$ inversion cost in full GP. We show the graphical model of prediction of SVGP in Figure 2.3. Furthermore, from the

perspective of weight space inference, the prediction of SVGP is equivalent to an approximate Bayesian linear model based on kernel feature map, $f(x) = \boldsymbol{w}^\top \Phi(\boldsymbol{x})$, where the following structured low-rank approximate posterior, parameterized by variational parameters $\boldsymbol{m_u}$ and $\boldsymbol{S_u}$, for $\boldsymbol{w}$ is considered.

$$q(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}; \boldsymbol{\Phi_Z^\top K_{ZZ}^{-1} m_u}, \boldsymbol{I} + \boldsymbol{\Phi_Z^\top K_{ZZ}^{-1}(S_u - K_{ZZ})K_{ZZ}^{-1}\Phi_Z}). \tag{2.37}$$

Given observed data $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{y})$, the inducing input locations $\boldsymbol{Z}$ are treated as model hyperparameters and are tuned jointly with the kernel hyperparameters and the variational parameters by ELBO maximization.

$$\begin{aligned}
\mathcal{L}_{ELBO} &= \mathbb{E}_{q(\boldsymbol{f},\boldsymbol{u},|\boldsymbol{Z},\boldsymbol{X})}[\log \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u},\boldsymbol{Z},\boldsymbol{X})p(\boldsymbol{u}|\boldsymbol{Z})}{q(\boldsymbol{f},\boldsymbol{u}|\boldsymbol{Z},\boldsymbol{X})}] \\
&= \int\int p(\boldsymbol{f}|\boldsymbol{u},\boldsymbol{Z},\boldsymbol{X})q(\boldsymbol{u}) \log \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{u}|\boldsymbol{Z})}{q(\boldsymbol{u})} d\boldsymbol{u} d\boldsymbol{f} \\
&= \underbrace{\mathbb{E}_{q(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{Z})}[\log p(\boldsymbol{y}|\boldsymbol{f})]}_{\text{ELL}} - \underbrace{\mathrm{KL}(q(\boldsymbol{u})||p(\boldsymbol{u}|\boldsymbol{Z}))}_{\text{KL regularizer}}
\end{aligned} \tag{2.38}$$

The full derivation of the above ELBO can be found in Appendix B.1.1. The VI framework also permits the use of non-Gaussian observation likelihood, which broadens the applications of GPs beyond regression tasks. For non-Gaussian likelihoods, the ELL term has no closed form, and we typically use Monte Carlo estimation, where the main cost ($O(M^3 + NM^2)$) comes from computing $q(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{Z})$. The KL regularizer can be evaluated analytically, since $q(\boldsymbol{u})$ and $p(\boldsymbol{u}|\boldsymbol{Z})$ are both Gaussian, and its main computational cost comes from the evaluation of the determinant $\det \boldsymbol{K_{ZZ}}$, and the matrix inversion $\boldsymbol{K_{ZZ}^{-1}}$ (both are $O(M^3)$). Combined, evaluating $\mathcal{L}_{ELBO}$ for SVGP incurs a cost of $O(M^3 + NM^2)$.

The posterior predictive distribution of $\boldsymbol{y}^*$ at test input locations, $\boldsymbol{X}^*$, is

$$\begin{aligned}
p(\boldsymbol{y}^*|\boldsymbol{X}^*,\boldsymbol{Z},\boldsymbol{u}) &= \int\int p(\boldsymbol{y}^*|\boldsymbol{f}^*)p(\boldsymbol{f}^*|\boldsymbol{u},\boldsymbol{Z},\boldsymbol{X^*})q(\boldsymbol{u})d\boldsymbol{u}d\boldsymbol{f}^* \\
&= \mathbb{E}_{q(\boldsymbol{f}^*|\boldsymbol{X}^*,\boldsymbol{Z})}[p(\boldsymbol{y^*}|\boldsymbol{f^*})].
\end{aligned} \tag{2.39}$$

Again, we can resort to Monte Carlo to estimate it in tasks requiring non-Gaussian observation likelihoods.

For regression based on Gaussian observation likelihood with noise variance σ^2, the ELL term can also be computed analytically. Thus, the optimal variational distribution $q_*(\boldsymbol{u})$ can be derived analytically, and it is a Gaussian,

$$q_*(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}; \frac{1}{\sigma^2}\boldsymbol{K_{ZZ}M^{-1}K_{ZX}y}, \boldsymbol{K_{ZZ}M^{-1}K_{ZZ}}), \tag{2.40}$$

where $\boldsymbol{M} := \boldsymbol{K_{ZZ}} + \frac{1}{\sigma^2}\boldsymbol{K_{ZX}}\boldsymbol{k_{XZ}}$. We commonly call the SVGP in this case sparse Gaussian process regression (SGPR; (Titsias, 2009)). Substituting $q_*(\boldsymbol{u})$ back into ELBO, we can obtain a so-called *collapsed variational bound*, which becomes an objective to further tune the kernel hyperparameters $\boldsymbol{\varphi}$ and inducing input locations $\boldsymbol{Z}$,

$$\begin{aligned}\mathcal{L}_{ELBO}(q_*) = \log \mathcal{N}(\boldsymbol{y}; \boldsymbol{0}, &\sigma^2\boldsymbol{I} + \boldsymbol{K_{ZX}}\boldsymbol{K_{ZZ}^{-1}}\boldsymbol{K_{ZX}}) \\ &- \frac{1}{2\sigma^2}\mathrm{Tr}(\boldsymbol{K_{XX}} - \boldsymbol{K_{ZX}}\boldsymbol{K_{ZZ}^{-1}}\boldsymbol{K_{ZX}}).\end{aligned} \tag{2.41}$$

Naturally, the corresponding posterior predictive distribution is also analytically tractable with a Gaussian form.

$$\begin{aligned}p(\boldsymbol{y}^*|\boldsymbol{X}^*, \mathcal{D}, \boldsymbol{Z}) = \mathcal{N}(\boldsymbol{y}^*; &\frac{1}{\sigma^2}\boldsymbol{K_{X^*Z}}\boldsymbol{M^{-1}}\boldsymbol{K_{ZX}}\boldsymbol{y}, \\ &\boldsymbol{K_{X^*X^*}} + \sigma^2\boldsymbol{I} \underbrace{-\boldsymbol{K_{X^*Z}}(\boldsymbol{K_{ZZ}^{-1}} - \boldsymbol{M}^{-1})^{-1}\boldsymbol{K_{ZX^*}}}_{\text{uncertainty reduction}}).\end{aligned} \tag{2.42}$$

For SVGP, the inducing points can be viewed as a compact summary of the training set, and the uncertainty reduction in the posterior predictive depends on the kernel similarity between $\boldsymbol{X}^*$ and $\boldsymbol{Z}$ (which tends to cover the whole region of the training inputs).

2.3.3 Deep Kernel Learning and Deep Gaussian Processes

It is not surprising that GPs may underperform in complex predictive tasks, compared with deep neural networks. After all, they are just shallow Bayesian linear models based on kernel feature as discussed previously. This section introduces two common approaches to enhance GPs with deep structures.

Deep kernel learning. While linear models rely on hand-crafted feature maps, deep neural networks can learn expressive feature representations for input data if we view the network right before the last layer as an expressive nonlinear feature extractor. Therefore, a natural way to improve GP is to combine kernel feature map and deep feature extractor. Deep kernel learning (Wilson et al., 2016) introduces deep kernel by parameterizing the kernel with a deep neural network. The network weights $\boldsymbol{\theta}$ then become hyperparameters of the deep kernel. Specifically, a deep kernel is obtained by compositing a regular base kernel $k_{base}(\cdot,\cdot)$ (e.g., RBF kernel) and a deep feature extractor parameterized by a deep neural network ($h_{\boldsymbol{\theta}}(\cdot)$): $k_{deep}(\cdot,\cdot) = k_{base}(h_{\boldsymbol{\theta}}(\cdot), h_{\boldsymbol{\theta}}(\cdot))$.

Deep Gaussian processes. In addition to incorporating deep structure in the kernel function, multiple GP layers, on their own, can also be stacked together to construct the so-called deep Gaussian processes (deep GPs) (Damianou and Lawrence, 2013). Unlike standard GPs, deep GPs are not liner models, and therefore the posterior inference becomes intractable. Again, we resort to the variational inducing points method discussed previously to approximate the posterior process for each GP layer (Salimbeni and Deisenroth, 2017). Consider an L-layer deep GP consisting of L latent functions $\{f_l\}_{l=1}^L$. The input for each layer, f_l, is the function value output by the previous layer: $\boldsymbol{f}_l = f_l(\boldsymbol{f}_{l-1})$. In each layer, a GP prior is used for its output: $p(f_l) = \mathcal{GP}(0(\cdot), k^{(l)}(\cdot,\cdot))$. Moreover, we also incorporate a set of inducing points $\{\boldsymbol{z}_l^{(i)}, u_l^{(i)}\}_{i=1}^{M_l}$ for each layer. The prior model is then

$$p(\boldsymbol{y}, \{\boldsymbol{f}_l\}_{l=1}^L, \{\boldsymbol{u}\}_{l=1}^L | \boldsymbol{X}, \{\boldsymbol{Z}_l\}_{l=1}^L) = \\ p(\boldsymbol{y}|\boldsymbol{f}_L)\left[\prod_{l=2}^{L} p(\boldsymbol{f}_l|\boldsymbol{u}_l, \boldsymbol{Z}_l, \boldsymbol{f}_{l-1}) p(\boldsymbol{u}_l|\boldsymbol{Z}_l)\right] p(\boldsymbol{f}_1|\boldsymbol{u}_1, \boldsymbol{Z}_1, \boldsymbol{X}) p(\boldsymbol{u}_1|\boldsymbol{Z}_1). \tag{2.43}$$

The approximate posterior of $\{\boldsymbol{u}_l\}_{l=1^L}$ is assumed to factorize between layers and the approximate posterior conditional distribution of $\boldsymbol{f}_l$ given $\boldsymbol{u}_l$, in each layer, is again assumed to be the same as the prior conditional distribution. In summary, the joint approximate posterior factorizes as follows:

$$q(\{\boldsymbol{f}_l\}_{l=1}^L, \{\boldsymbol{u}_l\}_{l=1}^L | \boldsymbol{X}, \{\boldsymbol{Z}_l\}_{l=1}^L) = \left[\prod_{l=2}^{L} p(\boldsymbol{f}_l|\boldsymbol{u}_l, \boldsymbol{Z}_l, \boldsymbol{f}_{l-1}) q(\boldsymbol{u}_l)\right] p(\boldsymbol{f}_1|\boldsymbol{u}_1, \boldsymbol{Z}_1, \boldsymbol{X}) q(\boldsymbol{u}_1). \tag{2.44}$$

The structured approximate posterior above allows us to obtain the following ELBO as training objective.

$$\mathcal{L}_{ELBO} = \mathbb{E}_{q(\boldsymbol{f}_L|\boldsymbol{X}, \{\boldsymbol{Z}_l\}_{l=1}^L)}[\log p(\boldsymbol{y}|\boldsymbol{f}_L)] - \sum_{l=1}^{L} \mathrm{KL}(q(\boldsymbol{u}_l)||p(\boldsymbol{u}_l|\boldsymbol{Z}_l)), \tag{2.45}$$

where

$$q(\boldsymbol{f}_L|\boldsymbol{X}, \{\boldsymbol{Z}_l\}_{l=1}^L) = \int \cdots \int q(\boldsymbol{f}_1|\boldsymbol{X}, \boldsymbol{Z}_1) \prod_{l=2}^{L} q(\boldsymbol{f}_l|\boldsymbol{f}_{l-1}, \boldsymbol{Z}_l) d\boldsymbol{f}_{1:L-1}. \tag{2.46}$$

The approximate posterior for the final output defined by the above integral is not analytically tractable, and in practice we can only work with samples from it. Usually, the final output samples from $q(\boldsymbol{f}_L|\boldsymbol{x}, \{\boldsymbol{Z}_l\}_{l=1}^L)$ is obtained by passing samples from $q(\boldsymbol{f}_l|\boldsymbol{f}_{l-1}, \boldsymbol{Z}_l)$ iteratively through each layer, which can be generated

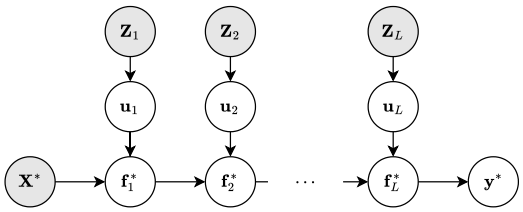


Figure 2.4: The graphical model of prediction of an L-layer deep Gaussian process with sparse GP approximations.

using the reparameterization trick since $q(\boldsymbol{f}_l|\boldsymbol{f}_{l-1}, \boldsymbol{Z}_l)$ is a Gaussian,

$$\begin{aligned} q(\boldsymbol{f}_l|\boldsymbol{f}_{l-1}, \boldsymbol{Z}_l) &= \int p(\boldsymbol{f}_l|\boldsymbol{u}_l, \boldsymbol{Z}_l, \boldsymbol{f}_{l-1}) q(\boldsymbol{u}_l) d\boldsymbol{u}_l \\ &= \mathcal{N}(\boldsymbol{K}_{\boldsymbol{f}_{l-1}\boldsymbol{Z}_l}^{(l)} \boldsymbol{K}_{\boldsymbol{Z}_l\boldsymbol{Z}_l}^{(l)-\mathbf{1}} \boldsymbol{m}_{\boldsymbol{u}_l}, \\ &\quad \boldsymbol{K}_{\boldsymbol{f}_{l-1}\boldsymbol{f}_{l-1}}^{(l)} + \boldsymbol{K}_{\boldsymbol{f}_{l-1}\boldsymbol{Z}_l}^{(l)} \boldsymbol{K}_{\boldsymbol{Z}_l\boldsymbol{Z}_l}^{(l)-\mathbf{1}} (\boldsymbol{S}_{\boldsymbol{u}_l} - \boldsymbol{K}_{\boldsymbol{Z}_l\boldsymbol{Z}_l}^{(l)}) \boldsymbol{K}_{\boldsymbol{Z}_l\boldsymbol{Z}_l}^{(l)-\mathbf{1}} \boldsymbol{K}_{\boldsymbol{Z}_l\boldsymbol{f}_{l-1}}^{(l)}). \end{aligned} \tag{2.47}$$

We show the graphical model of prediction of deep GP in Figure 2.4.

2.4 Deep Sequential Generative Models

In addition to predictive models, we also leverage probabilistic machine learning and deep neural networks to build deep generative models, which now form a rapidly evolving research field due to the tremendous advances from both topics. In short, deep generative models aim to estimate the true data distribution $p(\boldsymbol{x})$ with a parametric model $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ learned from a training set consisting of samples from $p(\boldsymbol{x})$, $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$, where $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is parameterized with deep neural networks with parameters $\boldsymbol{\theta}$. After learning, new samples can be generated from $p_{\boldsymbol{\theta}}(\boldsymbol{x})$, which can be treated as approximate samples from the true data-generating process. This section introduces the framework of deep sequential generative models, and we discuss in detail two popular models in this framework, hierarchical variational autoencoder and diffusion model.

2.4.1 Deep Latent Variable Models

We first review deep latent variable models (DLVMs; (Kingma and Welling, 2014; Rezende et al., 2014)) since many deep sequential generative models can be viewed as extensions of DLVMs. DLVMs assume that each observation $\boldsymbol{x}$ is generated by transforming a latent variable $\boldsymbol{z}$, through a deep neural network $f_{\boldsymbol{\theta}}(\cdot)$ with parameters $\boldsymbol{\theta}$. Typically, the distribution of the latent variable, $p_{\boldsymbol{\theta}}(\boldsymbol{z})$, is chosen to be a simple distribution (e.g., a standard Gaussian) a priori so that one can easily sample from it. We demonstrate the graphical model of a DLVM in Figure 2.5. Mathematically,

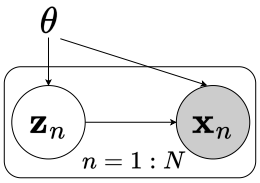


Figure 2.5: Graphical model of a deep latent variable model.

the generative model can be written as follows.

$$\boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z}) := \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \quad \boldsymbol{x} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) := \mathcal{N}(f_{\boldsymbol{\theta}}(\boldsymbol{z}), \sigma^2 \boldsymbol{I}). \tag{2.48}$$

Given a training set $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$, ideally, the model parameters $\boldsymbol{\theta}$ can be fitted by maximum likelihood estimation (MLE):

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n), \quad p_{\boldsymbol{\theta}}(\boldsymbol{x}_n) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{z}) d\boldsymbol{z}. \tag{2.49}$$

However, the MLE objective (Eq. 2.49) above requires solving complex integrals w.r.t. the latent variables $\boldsymbol{z}_n$ for the log marginal likelihoods $\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n)$, which are intractable since $f_{\boldsymbol{\theta}}(\boldsymbol{z})$ is a nonlinear deep neural network. In practice, we again resort to approximate inference and many DLVMs are learned via VI-based methods (Kingma and Welling, 2014; Rezende et al., 2014). We discuss a class of representative VI-based DLVMs, variational autoencoders, in detail.

Variational Autoencoders. Variational Autoencoders (VAEs) (Kingma and Welling, 2014; Rezende et al., 2014) are a class of DLVMs that employ VI for approximate maximum likelihood maximization. Notably, to reduce computational costs, they further consider amortized inference (Gershman and Goodman, 2014) to construct the approximate distributions in DLVMs: instead of constructing individual approximate distribution to the posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z}_n|\boldsymbol{x}_n)$ for each $\boldsymbol{x}_n$, VAEs use another deep neural network (commonly referred to as *inference network* or *encoder network*) to map each $\boldsymbol{x}_n$ to the variational parameters of its corresponding variational distribution. For example, when a factorized Gaussian distribution family, $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}), \boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\boldsymbol{x}))$, is considered, the variational mean and variance associated with an observation $\boldsymbol{x}$ are then obtained by an inference network $g_{\boldsymbol{\phi}}(\boldsymbol{x}) := [\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}), \log \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x})]$ with amortized variational parameters $\boldsymbol{\phi}$. For each observation $\boldsymbol{x}_n$, an ELBO can be constructed as a lower bound for the corresponding log marginal likelihood, and we can sum these ELBOs to form the final objective for joint training of the model

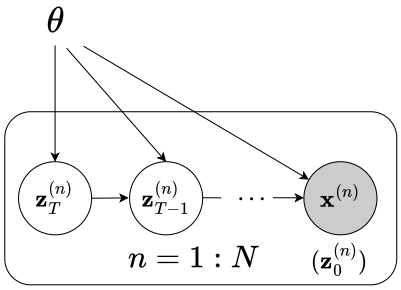


Figure 2.6: Graphical model of a deep sequential generative variable model.

parameters $\boldsymbol{\theta}$ and the variational parameters $\boldsymbol{\phi}$:

$$\begin{aligned} \boldsymbol{\theta}^*, \boldsymbol{\phi}^* &= \arg\max_{\boldsymbol{\theta},\boldsymbol{\phi}} \sum_{n=1}^{N} \mathcal{L}_{ELBO}(\boldsymbol{x}_n, \boldsymbol{\theta}, q_{\boldsymbol{\phi}}), \\ \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq \mathcal{L}_{ELBO}(\boldsymbol{x}, \boldsymbol{\theta}, q_{\boldsymbol{\phi}}) &= \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]}_{\text{ELL}} - \underbrace{\mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \| p_{\boldsymbol{\theta}}(\boldsymbol{z}))}_{\text{KL regularizer}}. \end{aligned} \tag{2.50}$$

The intractable expectation in the ELL term can again be approximated with Monte Carlo estimation and usually one sample is sufficient for decent empirical performance. To enable backpropagation through the inference network, the reparameterization trick (Kingma and Welling, 2014; Rezende et al., 2014) is again applied to sample from $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$:

$$\mathcal{L}_{ELBO}(\boldsymbol{x}, \boldsymbol{\theta}, q_{\boldsymbol{\phi}}) \approx \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}) + \boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{x}) \odot \boldsymbol{\epsilon}) - \mathrm{KL}(q_{\boldsymbol{\phi}} \| p_{\boldsymbol{\theta}}), \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}). \tag{2.51}$$

The inference network $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ and the DLVM's conditional distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ can be viewed as a stochastic encoder and decoder, respectively. This explains why we call the method an "autoencoder".

2.4.2 Hierarchical Variational Autoencoders

It can be challenging for DLVMs with only one latent variable to generate realistic observations, such as photo-realistic images, since they try to transform noise to target distribution in just one shot. Deep sequential generative models (DSGMs) extend DLVMs by introducing a sequence of latent variables and transforming noise into the target distribution using multiple intermediate steps. Specifically, they employ a sequence of latent variables $\boldsymbol{z}_1, \cdots, \boldsymbol{z}_T$ to capture representations of the data $\boldsymbol{z}_0 \coloneqq \boldsymbol{x}$ at different fidelity (Salimans, 2016):

$$\boldsymbol{x} = \boldsymbol{z}_0 \sim p_{\boldsymbol{\theta}}(\boldsymbol{z}_0) = \int p(\boldsymbol{z}_T) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) d\boldsymbol{z}_{1:T}, \tag{2.52}$$

where the prior distribution over the last latent variable $\boldsymbol{z}_T$ is often set to a simple distribution, such as standard Gaussian $p(\boldsymbol{z}_T) := \mathcal{N}(\boldsymbol{z}_T; \mathbf{0}, \boldsymbol{I})$, and the latent conditional distributions are parameterized by deep neural networks (we often only parameterize the conditional means):

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) := \mathcal{N}(\boldsymbol{z}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z}_t, t), \sigma_t^2 \boldsymbol{I}). \tag{2.53}$$

We show the graphical model of a DSGM in Figure 2.6.

Similar to DLVMs, direct MLE estimation for model parameters $\boldsymbol{\theta}$ of DSGMs is intractable, and again VI provides a principled framework for constructing a lower bound objective for log likelihood. A prominent VI-based DSGM is hierarchical variational autoencoders (HVAEs; (Sønderby et al., 2016; Maaløe et al., 2019; Vahdat and Kautz, 2020)). Specifically, HVAEs approximate the intractable posterior

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}_{1:T}|\boldsymbol{z}_0) = \frac{p(\boldsymbol{z}_T) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)}{p_{\boldsymbol{\theta}}(\boldsymbol{z}_0)} \tag{2.54}$$

with an amortized variational distribution (or inference model) $q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{z}_0)$. Different design choices for the factorization of the inference model have been proposed, including "bottom-up" factorization (Burda et al., 2015):

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{z}_0) = \prod_{t=1}^{T} q_{\boldsymbol{\phi}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}), \tag{2.55}$$

and "top-down" factorization (Sønderby et al., 2016):

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{z}_0) = q_{\boldsymbol{\phi}}(\boldsymbol{z}_T|\boldsymbol{z}_0) \prod_{t=1}^{T} q_{\boldsymbol{\phi}}(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{z}_0), \tag{2.56}$$

where the factors $q_{\boldsymbol{\phi}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$ and $q_{\boldsymbol{\phi}}(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{z}_0)$ are factorized Gaussian distributions with mean and diagnoal variance parameterized by neural networks. Given a set of training observations $\mathcal{D} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$ ($\boldsymbol{z}_0^{(n)} := \boldsymbol{x}^{(n)}$), we again jointly train both the generative model and the inference model by maximizing the sum of ELBOs:

$$\boldsymbol{\theta}^*, \boldsymbol{\phi}^* = \arg\max_{\boldsymbol{\theta}, \boldsymbol{\phi}} \sum_{n=1}^{N} \mathcal{L}_{ELBO}(\boldsymbol{x}^{(n)}, \boldsymbol{\theta}, q_{\boldsymbol{\phi}}),$$

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(n)}) \geq \mathcal{L}_{ELBO}(\boldsymbol{x}^{(n)}, \boldsymbol{\theta}, q_{\boldsymbol{\phi}}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}^{(n)}|\boldsymbol{z}_0^{(n)})} \left[\log \frac{p(\boldsymbol{z}_T^{(n)}) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1}^{(n)}|\boldsymbol{z}_t^{(n)})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}^{(n)}|\boldsymbol{z}_0^{(n)})} \right]. \tag{2.57}$$

Monte Carlo estimation and reparameterization trick can again be employed to further simplify the above objective and enable backpropagation.

2.4.3 Diffusion Models

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021b) are arguably the most successful DSGMs in recent years and they achieve photo-realistic image generation quality. It assumes a Markovian data-generating process (or denoising process) based on a sequence of latent variables as in Eq. 2.52 and the parameterization of the conditional distributions is similar to that in Eq. 2.53. However, unlike HVAEs, diffusion models define a fixed "bottom up" inference model (or diffusion process):

$$q(\boldsymbol{z}_{1:T}|\boldsymbol{z}_0) = \prod_{t=1}^{T} q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) = \prod_{t=1}^{T} \mathcal{N}(\boldsymbol{z}_t; \sqrt{\alpha_t}\boldsymbol{z}_{t-1}, (1-\alpha_t)\boldsymbol{I}), \quad \boldsymbol{z}_0 := \boldsymbol{x} \tag{2.58}$$

where $q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$'s are predefined Gaussian convolution kernels and no dimensionality reduction for latent variables is performed (i.e., $d = d_x$, where $d_x = \dim(\boldsymbol{x})$). Diffusion models are also trained by maximizing the ELBO but only with respect to the parameters $\boldsymbol{\theta}$ of the generation model, since the inference model is fixed and therefore does not include any trainable parameters. Since the inference model is simply defined to be a sequence of fixed Gaussian convolutions, one can analytically derive the corresponding "top-down" version of this inference model, which is of the form:

$$q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{z}_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}(\boldsymbol{z}_t, \boldsymbol{z}_0), \tilde{\beta}_t^2), \tag{2.59}$$

where $\tilde{\mu}$ and $\tilde{\beta}_t$ have analytical solutions with closed-form expressions. As a result of the simple and fixed inference model, the ELBO of diffusion models for each training observation $\boldsymbol{x}$ ($\boldsymbol{z}_0$) admits the following simplified form:

$$\begin{aligned}
\mathcal{L}_{ELBO}(\boldsymbol{x}, \boldsymbol{\theta}) &= \mathbb{E}_{q(\boldsymbol{z}_{1:T}|\boldsymbol{z}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{z}_0|\boldsymbol{z}_1) - \sum_{t=1}^{T} \mathrm{KL}(q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{z}_0)||p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}))\right] \\
&= \mathbb{E}_{q(\boldsymbol{z}_{1:T}|\boldsymbol{z}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{z}_0|\boldsymbol{z}_1) - \sum_{t=1}^{T} \underbrace{\frac{\|\mu_{\boldsymbol{\theta}}(\boldsymbol{z}_t, t) - \tilde{\mu}(\boldsymbol{z}_t, \boldsymbol{z}_0)\|^2}{2\sigma_t^2}}_{\text{mean matching}}\right].
\end{aligned} \tag{2.60}$$

The fixed inference model imposes strong inductive bias for the DSGM as the DSGM now is optimized towards a denoising process for a fixed diffusion process. The inductive bias can also be seen from the direct supervision signal for each intermediate latent variable in the "mean matching" term of the above ELBO. In practice, during each training iteration, we typically just sample one term from the above ELBO (consisting of $T+1$ terms) for each training observation to form a even simpler objective.

Based on the structure of $\tilde{\mu}(\boldsymbol{z}_t, \boldsymbol{z}_0)$, one can also reparameterize $\mu_{\boldsymbol{\theta}}(\boldsymbol{z}_t, t)$ accordingly to rewrite the "mean matching" term in other forms. For example, using the reparameterization trick, the noise parameterization of diffusion models (Ho et al., 2020; Song et al., 2021a) rewrites $\boldsymbol{z}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\boldsymbol{z}_t - \sqrt{\frac{1-\bar{\alpha}_t}{\bar{\alpha}_t}}\boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, where $\sqrt{\bar{\alpha}_t}$'s are fixed scalars determined by the fixed inference model. With it we can further reparameterize $\mu_{\boldsymbol{\theta}}(\boldsymbol{z}_t, t) = \tilde{\mu}(\boldsymbol{z}_t, \frac{1}{\sqrt{\bar{\alpha}_t}}\boldsymbol{z}_t - \sqrt{\frac{1-\bar{\alpha}_t}{\bar{\alpha}_t}}\epsilon_{\boldsymbol{\theta}}(\boldsymbol{z}_t, t))$, which allows us to rewrite the "mean matching" term into a "noise matching" term:

$$\lambda_t \|\epsilon_{\boldsymbol{\theta}}(\boldsymbol{z}_t, t) - \boldsymbol{\epsilon}\|^2 = \bar{\lambda}_t \|\epsilon_{\boldsymbol{\theta}}(\sqrt{\bar{\alpha}_t}\boldsymbol{z}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) - \boldsymbol{\epsilon}\|^2, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \tag{2.61}$$

where λ_t's are positive scalars determined by analytical form of $\tilde{\mu}$. This objective encourages the predicted noise $\epsilon_{\boldsymbol{\theta}}(\boldsymbol{z}_t, t)$ to match the noise $\boldsymbol{\epsilon}$ used to corrupt the observation $\boldsymbol{x}$ ($\boldsymbol{z}_0$) to its noisy version $\boldsymbol{z}_t$. Alternatively, by exploiting the connection between the denoising mean and the score of the noisy model, one can also obtain score-matching based objective to train diffusion models (Song et al., 2021b).

Chapter 3

Calibrating Transformers via Sparse Gaussian Processes

This chapter is based on Chen and Li (2023):

- Wenlong Chen and Yingzhen Li. **Calibrating Transformers via Sparse Gaussian Processes**. In *International Conference on Learning Representations (ICLR)*, 2023.

The main idea was developed by me while the last author provided useful suggestions. The manuscript was mainly written by me while the last author helped polish it. Moreover, I wrote all the code and performed all the experiments for this work.

In this chapter, we propose an approximate Bayesian inference method, sparse Gaussian process attention, to improve the uncertainty quantification in Transformer models. Generic weight-space inference techniques treat all deep neural networks as some parametric models and ignore the inductive bias within their architectures. Furthermore, because of the complex interactions between weights during the forward pass, there is no good reason to favor any specific prior or approximation techniques in these weight-space methods. In contrast, sparse Gaussian process attention is inspired by the identification of the connection between attention architecture in Transformers and the posterior mean of sparse variational Gaussian processes. This connection naturally informs us to use GP priors over the attention outputs and construct approximate posterior for them based on sparse GPs.

3.1 Introduction

Significant improvements have been made for deep sequence models in predictive tasks, and as reviewed in Section 2.1.1, Transformer (Vaswani et al., 2017), a deep sequence model architecture based on multi-head attention (MHA), has gained popularity in recent years. With Transformers being deployed in many downstream applications (Vaswani et al., 2017; Dosovitskiy et al., 2021; Brown et al., 2020), it is crucial to prevent poor robustness which often comes from erratic outputs with high confidence from these models (Guo et al., 2017; Mukhoti et al., 2020). This requires calibrated uncertainty quantification for Transformers which is much less well-studied at the time of this work, and it raises concerns about using Transformers for safety-critical tasks which require rational and risk-averse decision making under uncertainty.

Regarding uncertainty quantification, Bayesian inference is a powerful and principled framework to build probabilistic models for rational prediction and decision-making under uncertainty (Gal, 2016). Significant progress is observed for applying (approximate) Bayesian inference methods to quantify uncertainty in fully-connected, convolutional and recurrent neural networks (Blundell et al., 2015; Gal and Ghahramani, 2016; Zhang et al., 2019; Ritter et al., 2021). Initial efforts have been made on extending these techniques to Transformers but with mixed results (Tran et al., 2019; Xue et al., 2021). On the other hand, Gaussian processes (GPs), reviewed in Section 2.3, are gold standard methods for tasks requiring reliable function-space uncertainty estimates (Rasmussen and Williams, 2006; Wilson et al., 2020). Researchers have proposed to integrate deep learning ideas to GP model design, including deep kernel learning (Wilson et al., 2016) and deep GPs (Damianou and Lawrence, 2013; Salimbeni and Deisenroth, 2017). Still, these models have yet to be scaled to modern deep learning tasks such as large-scale image classification and language modeling.

In this work, we propose sparse Gaussian process attention (SGPA), a novel uncertainty quantification technique tailored to attention-based models (e.g., Transformers), by leveraging techniques from sparse variational Gaussian processes (SVGP) (Snelson and Ghahramani, 2005; Hensman et al., 2013) for improved uncertainty estimates. Our work presents the following insights and contributions:

- (Section 3.2) Our key observation is that kernel-based attention (Tsai et al., 2019) is equivalent to the posterior mean of an SVGP. This inspires us to specify GP priors for the attention outputs and leverage SVGP techniques for approximate posterior inference in Transformers. The resulting Transformer based on our SGPA approach can be viewed as a sparse deep GP (Salimbeni and Deisenroth, 2017) with deep kernel in use for each GP layers.

- (Section 3.3 & 3.4) We address the computational inefficiency issues of a naive extension of SVGP to multi-head self-attention with decoupled inducing points techniques (Cheng and Boots, 2017; Salimbeni et al., 2018), making SPGA scalable to deep learning tasks that Transformers are applied to.
- (Section 3.5) Empirically, on a variety of vision, NLP and graph prediction tasks and compared with baselines, SGPA-based Transformers improve considerably over in-distribution calibration, out-of-distribution (OOD) robustness, and OOD detection, while achieving competitive accuracy against Transformers with standard (Vaswani et al., 2017) or kernel attention (Tsai et al., 2019).

3.2 Sparse Gaussian Process Attention

We propose Sparse Gaussian Process Attention (SGPA) to perform approximate Bayesian inference for Transformer-based models. The key idea is to replace the softmax operation in scaled dot-product attention (Vaswani et al., 2017) with a kernel (Tsai et al., 2019), and connect the resulting attention to the mean of an SVGP. This insight allows us to apply SVGP equations for uncertainty estimation, and we further introduce decoupled inducing points to improve computational efficiency.

3.2.1 Attention as the Mean of a Sparse Variational Gaussian Process

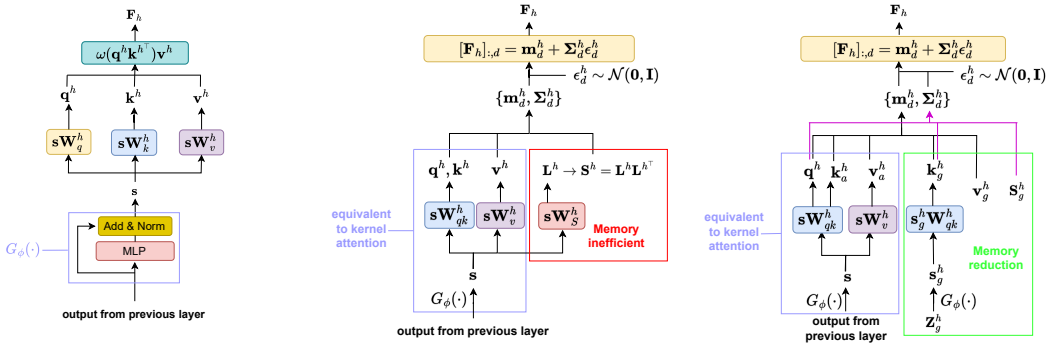
Standard Transformers use attention blocks based on scaled dot-product (Vaswani et al., 2017). Given queries $\boldsymbol{q} \in \mathbb{R}^{T \times d_q}$, keys $\boldsymbol{k} \in \mathbb{R}^{T \times d_k}$ and value $\boldsymbol{v} \in \mathbb{R}^{T \times d_v}$, the scaled dot-product (SDP) attention is given as follows:

$$\text{SDP-Attention:} \quad \boldsymbol{F} = \text{softmax}(\frac{\boldsymbol{q}\boldsymbol{k}^\top}{\sqrt{d_k}})\boldsymbol{v}, \tag{3.1}$$

where d_k $(= d_q)$ is the dimension of the keys. Since attention involves measuring the similarity between $\boldsymbol{q}$ and $\boldsymbol{k}$, Tsai et al. (2019) generalized SDP-Attention by replacing $\text{softmax}(\frac{\boldsymbol{q}\boldsymbol{k}^\top}{\sqrt{d_k}})$ in Eq. 3.1 with a kernel gram matrix $\boldsymbol{K}_{\boldsymbol{qk}}$ $([\boldsymbol{K}_{\boldsymbol{qk}}]_{i,j} = k(\boldsymbol{q}_i, \boldsymbol{k}_j))$ computed using a valid symmetric kernel $k(\cdot, \cdot)$, for which we refer to it as kernel attention or K-Attention for short:

$$K\text{-Attention:} \quad \boldsymbol{F} = \boldsymbol{K}_{\boldsymbol{qk}}\boldsymbol{v}. \tag{3.2}$$

Recall that the posterior mean of SVGP in Eq. 2.36 reviewed in the Section 2.3.2 is $\boldsymbol{m} = \boldsymbol{K}_{\boldsymbol{XZ}}\boldsymbol{K}_{\boldsymbol{ZZ}}^{-1}\boldsymbol{m}_{\boldsymbol{u}}$, when evaluated on inputs $\boldsymbol{X}$. Now we reparameterize the variational mean parameter of SVGP as $[\boldsymbol{v}]_{:,d} := \boldsymbol{K}_{\boldsymbol{ZZ}}^{-1}\boldsymbol{m}_{\boldsymbol{u}}$ for each dimension (d) of



(a) Vanilla Transformer (b) Standard SGPA (ours) (c) Decoupled SGPA (ours)

Figure 3.1: Illustration of one head (h) of multi-head self attention in one layer of (a) vanilla Transformer, (b) Transformer based on standard SGPA and (c) Transformer based on decoupled SGPA.

$\boldsymbol{v}$, and define the queries and keys as the input locations and inducing point locations: $\boldsymbol{q} := \boldsymbol{X}, \boldsymbol{k} := \boldsymbol{Z}$. This allows us to rewrite the posterior mean of SVGP as

$$\text{SVGP-mean} \quad \boldsymbol{m} = \boldsymbol{K_{qk}}[\boldsymbol{v}]_{:,d} \tag{3.3}$$

By doing so, equivalence can be identified between the posterior mean of an SVGP and each dimension of the output of a kernel attention block. This allows us to naturally extend the toolbox of Gaussian processes and their scalable approximations for quantifying uncertainty in Transformers in the following sections.

3.2.2 Standard SGPA & its Inefficiency for Self-attention

Observing the equivalence between K-Attention and SVGP mean, a natural idea for uncertainty estimation is to apply SVGP techniques for approximate posterior variance computations. In detail, we introduce a set of variational covariance parameters $\boldsymbol{S} \in \mathbb{R}^{T \times T \times d_v}$ (with T as the number of keys/inducing inputs), and optimize them using the ELBO (Eq. 2.38) as discussed in Section 2.3.2 (see Appendix B.1.1 for full derivation). This procedure returns the mean and covariance for each dimension (d) of the posterior attention output as:

$$\boldsymbol{m}_d = \boldsymbol{K_{qk}}[\boldsymbol{v}]_{:,d}, \quad \boldsymbol{\Sigma}_d = \boldsymbol{K_{qq}} + \boldsymbol{K_{qk}}(\boldsymbol{K_{kk}^{-1}}[\boldsymbol{S}]_{:,:,d}\boldsymbol{K_{kk}^{-1}} - \boldsymbol{K_{kk}^{-1}})\boldsymbol{K_{kq}}. \tag{3.4}$$

In this way, we fit an SVGP for each dimension of attention outputs independently: for each dimension d, an SVGP given in Eq. 3.4 is fitted using the same kernel, but with different variational mean ($[\boldsymbol{v}]_{:,d}$) and covariance ($[S]_{:,:,d}$) parameters. We name this approach as *standard SGPA* and provide a visualization of the operations in Figure 3.1b.

Unfortunately, standard SGPA becomes computationally inefficient when applied to Transformers based on multi-head self-attention. In each attention layer the keys for each head h, $\boldsymbol{k}^h$, are obtained by passing the output from previous layer through a neural network. Moreover, the projection matrices (defined in Section 2.1.1) for queries and keys are tied as in Tsai et al. (2019) (i.e., $\boldsymbol{W}_q^h = \boldsymbol{W}_k^h := \boldsymbol{W}_{qk}^h \in \mathbb{R}^{d_s \times d_k}$) to obtain a valid symmetric kernel. As a result, the queries and keys in a self-attention layer are the same, more importantly they are input-dependent, i.e., $\boldsymbol{k}^h = \boldsymbol{s}\boldsymbol{W}_k^h$ (where $\boldsymbol{s} \in \mathbb{R}^{T \times d_s}$ is the input to the MHSA block) and they vary as the input sequence to the Transformer changes. Therefore, to extend standard SVGP framework (Eq. 3.4) to self-attention, the covariance parameters $\boldsymbol{S}^h$ need to be input-dependent as well to accommodate the varying inducing inputs $\boldsymbol{k}^h$. A naive idea would parameterize $\boldsymbol{S}^h$ by linear projection, e.g., $vec(\boldsymbol{L}^h) = \boldsymbol{s}\boldsymbol{W}_s^h$ for one head where $\boldsymbol{L}^h$ is the Cholesky factor of $\boldsymbol{S}^h$ (see Figure 3.1b). This will incur a memory cost of $O(T^2)$ per head even if we tie the variational covariances across output dimensions, and a run-time cost of $O(T^3)$ per head per input sequence for inverting $\boldsymbol{K}_{\boldsymbol{k}^h\boldsymbol{k}^h}$ as $\boldsymbol{k}^h$ is input-dependent. Therefore standard SGPA is both memory and run-time inefficient especially for long input sequences.

Remark 3.2.1. Unlike the case where variational parameters are freely-optimized variables, note that in self-attention settings, different parameterizations of the posterior mean in general define different posterior predictive distributions. To see this, suppose that the value for one head in a MHSA block $\boldsymbol{v}^h$ is obtained by projection $\boldsymbol{v}^h = \boldsymbol{s}^h\boldsymbol{W}_v^h$ ($\boldsymbol{W}_v^h \in \mathbb{R}^{d_s \times d_v}$) and the reparameterized SVGP posterior mean yields $\boldsymbol{m}^h = \boldsymbol{K}_{q^hk^h}\boldsymbol{v}^h = \boldsymbol{K}_{q^hk^h}\boldsymbol{s}^h\boldsymbol{W}_v^h$. Now given the canonical posterior mean parameterization and the corresponding values obtained via $\tilde{\boldsymbol{v}}^h = \boldsymbol{s}^h\tilde{\boldsymbol{W}}_v^h$, in order to obtain the same posterior mean as the reparameterized one for all N sequences in the dataset, we need to solve for $\tilde{\boldsymbol{W}}_v^h$ with N linear systems of the form

$$\boldsymbol{K}_{q^hk^h}\boldsymbol{K}_{k^hk^h}^{-1}\boldsymbol{s}^h\tilde{\boldsymbol{W}}_v^h = \boldsymbol{K}_{q^hk^h}\boldsymbol{s}^h\boldsymbol{W}_v^h \iff \left(\boldsymbol{K}_{k^hk^h}^{-1}\boldsymbol{s}^h\right)\tilde{\boldsymbol{W}}_v^h = \boldsymbol{s}^h\boldsymbol{W}_v^h. \tag{3.5}$$

Hence, we need to solve a linear system with $N \times T \times d_v$ equations in total, and $d_s \times d_v$ variables. In general, since $N \times T \gg d_s$, it does not have a solution.

3.3 Improving Time & Memory Efficiencies via Decoupled SGPA

We propose to address the aforementioned inefficiency issues by extending the decoupled sparse Gaussian process approximation (Cheng and Boots, 2017; Salimbeni et al., 2018) to self-attention SGPA.

3.3.1 Preliminaries of Decoupled SVGPs

Decoupled SVGPs (Cheng and Boots, 2017; Salimbeni et al., 2018) can be interpreted as SVGPs but with structured variational distributions for the inducing points. Suppose we split the inducing points into two sets: $\{(\boldsymbol{z}_g^{(m)}, u_g^{(m)})\}_{m=1}^{M_g}$ and $\{(\boldsymbol{z}_a^{(m)}, u_a^{(m)})\}_{m=1}^{M_a}$. The original decoupled SVGP (DSVGP; (Cheng and Boots, 2017)) considers a structured variational distribution over $\boldsymbol{u} := \boldsymbol{u}_{g\cup a} = \boldsymbol{u}_g \cup \boldsymbol{u}_a$:

$$q(\boldsymbol{u}) = q(\boldsymbol{u_g})q(\boldsymbol{u}_a|\boldsymbol{u}_g) = \mathcal{N}(\boldsymbol{u}_g; \boldsymbol{0}, \boldsymbol{S}_g)\mathcal{N}(\boldsymbol{u}_a; \boldsymbol{C}\boldsymbol{u}_g + \boldsymbol{m}_a, , \boldsymbol{B}), \tag{3.6}$$

where $\boldsymbol{B} = \boldsymbol{K}_{\boldsymbol{Z}_a\boldsymbol{Z}_a} - \boldsymbol{K}_{\boldsymbol{Z}_a\boldsymbol{Z}_g}\boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_g}^{-\boldsymbol{1}}\boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_a}$ and $\boldsymbol{C} = \boldsymbol{K}_{\boldsymbol{Z}_a\boldsymbol{Z}_g}\boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_g}^{-\boldsymbol{1}}$. The corresponding variational mean and covariance are:

$$\boldsymbol{m_u} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{m}_a \end{pmatrix}, \quad \boldsymbol{S_u} = \begin{pmatrix} \boldsymbol{S}_g & \boldsymbol{S}_g\boldsymbol{C}^\top \\ \boldsymbol{C}\boldsymbol{S}_g & \boldsymbol{B} + \boldsymbol{C}\boldsymbol{S}_g\boldsymbol{C}^\top \end{pmatrix}. \tag{3.7}$$

Some terms used in the posterior predictive covariance of SVGP can then be canceled out:

$$\boldsymbol{K}_{\boldsymbol{ZZ}}^{-1}\boldsymbol{S_u} = \begin{bmatrix} (\boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_g}^{-1}\boldsymbol{S}_g - \boldsymbol{I})\boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_g}^{-1}\boldsymbol{K}_{\boldsymbol{Z}_a\boldsymbol{Z}_g} & \boldsymbol{K}_{\boldsymbol{Z}_a\boldsymbol{Z}_a}^{-1}\boldsymbol{S}_g \\ \boldsymbol{I} & \boldsymbol{0} \end{bmatrix}, \tag{3.8}$$

$$\boldsymbol{K}_{\boldsymbol{ZZ}}^{-1}\boldsymbol{S}_u\boldsymbol{K}_{\boldsymbol{ZZ}}^{-1} - \boldsymbol{K}_{\boldsymbol{ZZ}}^{-1} = \begin{bmatrix} \boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_g}^{-1}\boldsymbol{S}_g\boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_g}^{-1} - \boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_g}^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}. \tag{3.9}$$

Plugging this structured variational mean and covariance parameters into the posterior predictive distribution of SVGP (Eq. 2.36) allows us to obtain the posterior distribution $q(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{Z})$ used in DSVGP, which is a Gaussian with the following mean $\boldsymbol{m_f}$ and covariance $\boldsymbol{\Sigma_{ff}}$:

$$\begin{aligned} \boldsymbol{m_f} &= \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{Z}_a}\boldsymbol{K}_{\boldsymbol{Z}_a\boldsymbol{Z}_a}^{-1}\boldsymbol{m}_a, \\ \boldsymbol{\Sigma_{ff}} &= K_{\boldsymbol{XX}} + \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{Z}_g}\boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_g}^{-1}(\boldsymbol{S}_g - \boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_g})\boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_g}^{-1}\boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{X}}. \end{aligned} \tag{3.10}$$

Clearly, the inducing points used for this posterior predictive mean and covariance are now completely decoupled. Moreover, since in the posterior predictive distribution $q(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{Z})$, the variational mean parameter $\boldsymbol{m}_a$ always appears together with a left matrix multiplication of $\boldsymbol{K}_{\boldsymbol{Z}_a\boldsymbol{Z}_a}^{-1}$, we can reparameterize the variational mean as $\boldsymbol{v}_a := \boldsymbol{K}_{\boldsymbol{Z}_a\boldsymbol{Z}_a}^{-1}\boldsymbol{m}_a$, which is then treated as a final variational parameter to optimize. As a result, the posterior predictive mean can be rewritten as $\boldsymbol{m_f} = \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{Z}_a}\boldsymbol{v}_a$.

Orthogonally decoupled SVGP (ODSVGP; (Salimbeni et al., 2018)) considers another

structured Gaussian variational distribution for the two sets of inducing points:

$$\begin{aligned} q(\boldsymbol{u}) &= q(\boldsymbol{u_g})q(\boldsymbol{u}_a|\boldsymbol{u}_g) \\ &= \mathcal{N}(\boldsymbol{u}_g; \boldsymbol{m}_g, \boldsymbol{S}_g)\mathcal{N}(\boldsymbol{u}_a; \boldsymbol{C}\boldsymbol{u}_g + \boldsymbol{B}\boldsymbol{K}_{\boldsymbol{Z_a}\boldsymbol{Z_a}}^{-1}\boldsymbol{m}_a, \boldsymbol{B}), \end{aligned} \tag{3.11}$$

The corresponding variational mean and covariance become:

$$\boldsymbol{m_u} = \begin{pmatrix} \boldsymbol{m}_g \\ \boldsymbol{C}\boldsymbol{m}_g + \boldsymbol{B}\boldsymbol{K}_{\boldsymbol{Z}_a\boldsymbol{Z}_a}^{-1}\boldsymbol{m}_a \end{pmatrix}, \quad \boldsymbol{S_u} = \begin{pmatrix} \boldsymbol{S}_g & \boldsymbol{S}_g\boldsymbol{C}^\top \\ \boldsymbol{C}\boldsymbol{S}_g & \boldsymbol{B} + \boldsymbol{C}\boldsymbol{S}_g\boldsymbol{C}^\top \end{pmatrix}. \tag{3.12}$$

Again, plugging the above $\boldsymbol{m_u}$ and $\boldsymbol{S_u}$ in Eq. 2.36, we can obtain the posterior predictive distribution of ODSVGP for $\boldsymbol{f}$ after canceling some terms (Eq. 3.8 and 3.9). It is a Gaussian with mean $\boldsymbol{m_f}$ and covariance $\boldsymbol{\Sigma_{ff}}$ given as:

$$\begin{aligned} \boldsymbol{m_f} &= \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{Z}_a}\boldsymbol{K}_{\boldsymbol{Z}_a\boldsymbol{Z_a}}^{-1}\boldsymbol{m}_a - \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{Z}_g}\boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_g}^{-1}\boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_a}\boldsymbol{K}_{\boldsymbol{Z}_a\boldsymbol{Z}_a}^{-1}\boldsymbol{m}_a + \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{Z}_g}\boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_g}^{-1}\boldsymbol{m}_g, \\ \boldsymbol{\Sigma_{ff}} &= K_{\boldsymbol{X}\boldsymbol{X}} + \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{Z}_g}\boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_g}^{-1}(\boldsymbol{S}_g - \boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_g})\boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_g}^{-1}\boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{X}}. \end{aligned} \tag{3.13}$$

By again reparameterizing the variational mean parameters, $\boldsymbol{v}_a := \boldsymbol{K}_{\boldsymbol{Z}_a\boldsymbol{Z}_a}^{-1}\boldsymbol{m}_a$ and $\boldsymbol{v}_g := \boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_g}^{-1}\boldsymbol{m}_g$, we arrive at the final expressions of the posterior predictive mean and covariance of ODSVGP:

$$\begin{aligned} \boldsymbol{m_f} &= \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{Z}_a}\boldsymbol{v}_a - \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{Z}_g}\boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_g}^{-1}\boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_a}\boldsymbol{v}_a + \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{Z}_g}\boldsymbol{v}_g, \\ \boldsymbol{\Sigma_{ff}} &= K_{\boldsymbol{X}\boldsymbol{X}} + \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{Z}_g}\boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_g}^{-1}(\boldsymbol{S}_g - \boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_g})\boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{Z}_g}^{-1}\boldsymbol{K}_{\boldsymbol{Z}_g\boldsymbol{X}}. \end{aligned} \tag{3.14}$$

Notice that unlike DSVGP, the posterior predictive mean of ODSVGP are computed based on both sets of inducing points. Through our preliminary experiments at the stage of hyperparameter tuning, we find ODSVGP works better than DSVGP when applied in the SGPA framework (see Appendix B.3.1), so we focus on integrating ODSVGP in SGPA and call the resulting method decoupled SGPA from now on.

3.3.2 Decoupled SGPA

When applying ODSVGP in the framewok of SGPA, in addition to input-dependent (or "amortized") keys/inducing inputs $\boldsymbol{k}^h$, which we will call $\boldsymbol{k}_a^h$ from now on, for each head h, we also incorporate another M_g number of "global" keys/inducing inputs $\boldsymbol{k}_g^h$ that are shared across all input sequences. The main idea is to compute the variance of sparse GP using the global keys only, so that the variational parameters for the $\boldsymbol{S}^h$ matrix become independent to the input sequences. Indeed, following Eq. 3.14 presented in the above paragraph, we can compute the mean and covariance for each output dimension (d) of each head as (we drop the superscript h here for more

Table 3.1: Complexity comparison for standard and decoupled SGPA.

Model	Time	Additional Memory
MLE	$O(BT^2)$	-
Standard SGPA	$O(BT^3)$	$O(T^2)$
Decoupled SGPA	$O(BT^2M_g + M_g^3)$	$O(M_g^2)$

concise notation):

$$\begin{aligned}\boldsymbol{m}_d &= \boldsymbol{K}_{\boldsymbol{q}\boldsymbol{k}_a}[\boldsymbol{v}_a]_{:,d} - \boldsymbol{K}_{\boldsymbol{q}\boldsymbol{k}_g}\boldsymbol{K}_{\boldsymbol{k}_g\boldsymbol{k}_g}^{-1}\boldsymbol{K}_{\boldsymbol{k}_g\boldsymbol{k}_a}[\boldsymbol{v}_a]_{:,d} + \boldsymbol{K}_{\boldsymbol{q}\boldsymbol{k}_g}[\boldsymbol{v}_g]_{:,d},\\ \boldsymbol{\Sigma}_d &= \boldsymbol{K}_{\boldsymbol{q}\boldsymbol{q}} + \boldsymbol{K}_{\boldsymbol{q}\boldsymbol{k}_g}\boldsymbol{K}_{\boldsymbol{k}_g\boldsymbol{k}_g}^{-1}([\boldsymbol{S}_g]_{:,:,d} - \boldsymbol{K}_{\boldsymbol{k}_g\boldsymbol{k}_g})\boldsymbol{K}_{\boldsymbol{k}_g\boldsymbol{k}_g}^{-1}\boldsymbol{K}_{\boldsymbol{k}_g\boldsymbol{q}},\end{aligned} \tag{3.15}$$

where $\boldsymbol{v}_g \in \mathbb{R}^{M_g \times d_v}$, $\boldsymbol{S}_g \in \mathbb{R}^{M_g \times M_g \times d_v}$ are the variational parameters associated with the global keys $\boldsymbol{k}_g$, and $\boldsymbol{v}_a \in \mathbb{R}^{T \times d_v}$ is computed via projection $\boldsymbol{v}_a = \boldsymbol{s}\boldsymbol{W}_v$. We name this approach as *decoupled SPGA* which is illustrated in Figure 3.1c.

Compared to standard SGPA (Eq. 3.4), where $\boldsymbol{k}_a^h$ in decoupled SGPA is the same as $\boldsymbol{k}^h$ in standard SGPA), we see that the posterior mean of decoupled SGPA also involves two extra terms to take into account the effect of global inducing points. But more importantly, the posterior variance of the two SGPA methods differ only in the keys/inducing inputs in use (input-dependent keys $\boldsymbol{k}^h$ versus global keys $\boldsymbol{k}_g^h$), and this brings in the key advantange of decoupled SGPA. As the posterior covariance in Eq. 3.15 only involves the global inducing points, the variational covariance no longer needs to be input-dependent, and (the Cholesky factor of) $\boldsymbol{S}_g^h$ can be parameterized freely. Now the number of parameters for the covariance part is of order of $O(M_g^2)$ (vs $O(T^2)$ in standard SPGA), and the computation of matrix inversion pays a one-off cost of $O(M_g^3)$ (vs $O(T^3)$ for every input sequence). Notice that we are free to choose the number of global inducing points M_g, and in practice we find $M_g = O(\frac{T_{avg}}{H})$ is usually sufficient for experiments considered in this work, where T_{avg} is the average length of training input sequences. In Table 3.1, we summarize time complexity (with batch size B) and the additional memory (number of parameters) required for SGPA in one head of a Transformer. We also include maximum likelihood estimation (MLE) for reference (note that memory complexity for MLE does not depend on input sequence length T). As the time and memory savings are significant, we mainly evaluate decoupled SGPA in our experiments, and in the rest of the main text we will refer to decoupled SGPA as SGPA for short.

3.4 Transformer Based on Decoupled SGPA

So far we have presented SGPA methods for uncertainty quantification in a multi-head self-attention module. When applied to Transformer models, multiple layers of

attention blocks are in use, and in the following we describe the construction of a Transformer model based on decoupled SGPA. Note that, as SGPA is equivalent to a sparse GP, the Transformer model presented below can be viewed as a sparse approximation to a deep GP (Damianou and Lawrence, 2013; Salimbeni and Deisenroth, 2017) with deep kernel in each layer.

Our Transformer architecture mostly follows the one in Vaswani et al. (2017). The input to the l-th SGPA layer is the output from previous SGPA layer $\boldsymbol{F}^{l-1} \in \mathbb{R}^{T \times d^{l-1}}$. We first process the input with a non-linear mapping $G_{\phi^l} : \mathbb{R}^{d^{l-1}} \to \mathbb{R}^{d^l}$, and then perform projections to obtain the queries, amortized & global keys and values. Specifically, we have for each head h:

$$\boldsymbol{q}^{l,h} = \boldsymbol{k}_a^{l,h} = G_{\phi^l}(\boldsymbol{F}^{l-1})\boldsymbol{W}_{qk}^{l,h}, \quad \boldsymbol{k}_g^{l,h} = G_{\phi^l}(\boldsymbol{Z}_g^{l,h})\boldsymbol{W}_{qk}^{l,h}, \quad \boldsymbol{v}_a^{l,h} = G_{\phi^l}(\boldsymbol{F}^{l-1})\boldsymbol{W}_{v}^{l,h}, \tag{3.16}$$

where $\boldsymbol{Z}_g^{l,h} \in \mathbb{R}^{M_g \times d^{l-1}}$ are global inducing locations of the l-th layer defined on the same space as $\boldsymbol{F}^{l-1}$. Then we apply a base kernel $K_{base}(\cdot, \cdot)$ to compute the kernel matrices. This is equivalent to using a deep kernel defined on the space of $\boldsymbol{F}^{l-1}$, and the parameters of G_{ϕ^l} are viewed as the hyperparameters of the deep kernel. Lastly, with variational parameters $(\boldsymbol{v}_g^{l,h}, \boldsymbol{S}_g^{l,h})$ associated with the global inducing locations $\boldsymbol{Z}_g^{l,h}$, we can obtain $\boldsymbol{m}_d^{l,h}$ and $\boldsymbol{\Sigma}_d^{l,h}$ using Eq. 3.15. We then propagate uncertainty to the next layer by generating samples of output for each head using the reparameterization trick as in Salimbeni and Deisenroth (2017):

$$[\boldsymbol{F}_h^l]_{:,d} = \boldsymbol{m}_d^{l,h} + \boldsymbol{L}_{\boldsymbol{\Sigma}_d}^{l,h} \boldsymbol{\epsilon}_d^{l,h}, \quad \boldsymbol{\epsilon}_d^{l,h} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \tag{3.17}$$

where $\boldsymbol{L}_{\boldsymbol{\Sigma}_d}^{l,h}$ is the Cholesky factor of $\boldsymbol{\Sigma}_d^{l,h}$. The final output $\boldsymbol{F}^l \in \mathbb{R}^{T \times d^l}$ of this SGPA layer is obtained by linear combination in the same way as in standard Transformers (see Eq. 2.4).

The ELBO objective for training the variational & kernel parameters is derived following deep GP and additive GP (Duvenaud et al., 2011) approaches. The key idea is that as each head in MHSA with SGPA is a (sparse) GP, the final output $\boldsymbol{F}^l$ can also be viewed as a weighted summation of (sparse) GPs, which is again a GP (Duvenaud et al., 2011). This allows us to perform variational approximations on each of the heads before the final combination instead of using a direct approximation on the $\boldsymbol{F}^l$ process (Sun et al., 2021). Assuming the approximate posterior q for $\{\boldsymbol{F}_h^l\}_{h=1}^H$ factorizes over h, the corresponding ELBO with input sequence $\boldsymbol{F}^0 := \boldsymbol{X}$ is

(derivations in Appendix B.1.2):

$$\mathcal{L}_{ELBO} = \mathbb{E}_{q(\boldsymbol{F}^L|\boldsymbol{F}^0,\{\boldsymbol{k}_g^{l,h}\}_{l=1,h=1}^{L,H})}[\log p(\boldsymbol{Y}|\boldsymbol{F}^L)] - \sum_{l=1}^{L}\sum_{h=1}^{H}\mathbb{E}_{q(\boldsymbol{F}^l|\boldsymbol{F}^0,\{\boldsymbol{k}_g^{j,h}\}_{j=1,h=1}^{l,H}))}[\mathrm{KL}(q(\boldsymbol{u}_{a\cup g}^{l,h}|\boldsymbol{k}_g^{l,h},\boldsymbol{F}^{l-1})||p(\boldsymbol{u}_{a\cup g}^{l,h}|\boldsymbol{k}_g^{l,h},\boldsymbol{F}^{l-1}))]. \tag{3.18}$$

In practice, we resort to Monte-Carlo to estimate $\mathcal{L}_{ELBO}$ with samples of function values generated iteratively while passing through each layer using the reparameterization trick (Eq. 3.17).

3.5 Experiments

We evaluate SGPA on prediction tasks across modalities, with the following experimental set-up.

- Datasets: CIFAR10 & CIFAR100 (image classification (Krizhevsky et al., 2009), CV tasks); CoLA (linguistic acceptability prediction (Warstadt et al., 2019), NLP task) and IMDB (sentiment analysis, (Maas et al., 2011), NLP task).
- Network architectures: We use Vision Transformers (ViT (Dosovitskiy et al., 2021)) for CV tasks. For kernel attention we use the exponential kernel (Tsai et al., 2019) and the ARD-RBF kernel (Rasmussen and Williams, 2006) for NLP and CV tasks respectively. Scaled dot-product (SDP) attention based Transformers are also evaluated. As in Tsai et al. (2019), we find kernel attention tends to outperform SDP attention in most tasks considered, thus we do not include the results of SDP attention in the main text. These results can be found in the tables in Appendix B.7.
- Baselines: We compare our approach with the following "single-model" methods: maximum likelihood estimation (MLE), Bayesian inference methods including mean-field variational inference (MFVI, (Blundell et al., 2015)), Monte-Carlo Dropout (MCD, (Gal and Ghahramani, 2016)), Kronecker-factored last layer Laplace approximation (KFLLLA) (Kristiadi et al., 2020), and Spectral-normalized Neural Gaussian Process (SNGP) (Liu et al., 2020). For tasks where a validation set is used, we also consider temperature scaling (TS) (Guo et al., 2017) and use the validation set as the calibration set. For CV tasks, we also consider ensemble methods: we compare SGPA ensemble (SGPAE) with deep ensemble (DE) (Lakshminarayanan et al., 2017). We don't consider ensemble models in NLP tasks since we use different train-(valid)-test splits in different runs for them.

- Evaluations & metrics: We consider three evaluation set-ups: in-distribution performance, out-of-distribution (OOD) robustness and OOD detection. The metrics on test set include predictive accuracy metrics for each task, uncertainty calibration metrics such as negative predictive log-likelihood (NLL), expected calibration error (ECE) and maximum calibration error (MCE) (Naeini et al., 2015; Guo et al., 2017). A brief introduction for these uncertainty calibration metrics is provided in Appendix B.2. We report the mean±two standard errors for each metric obtained from 5 independent runs. For OOD detection tasks we consider the area under the ROC & precision-recall curves (AUROC & AUPR, respectively), and we report the average ranks in terms of AUROC and AUPR over all of the 6 OOD detection tasks for each method.

For fair comparisons, within each task, all the models are trained using the same architecture and optimization setting. All the models are trained from scratch without pre-training. We include the experimental details in Appendix B.4 and the comparison of wall-clock running time in Appendix B.5. Results in tables are also presented in Appendix B.7.

3.5.1 In-distribution Calibration

We report the evaluation results for in-distribution test data on image classification (CIFAR10 & CIFAR100, without data augmentation), sentiment analysis (IMDB), and linguistic acceptability (CoLA) tasks in the first, second, third and fourth row of Figure 3.2 respectively. Here for the CoLA dataset, predictive accuracy is measured by Matthew correlation coefficient (MCC) (Matthews, 1975) instead of accuracy, as in Warstadt et al. (2019).

All "single-model" calibration methods considered tend to improve the calibration, except for sentiment analysis, where KFLLLA fails in the sense that it achieves worse calibration even than MLE (although KFLLLA achieves best calibration for linguistic acceptability (CoLA), its performance is unstable across tasks). Although MFVI tends to achieve the lowest calibration errors, it severely underfits the data in all the experiments. This is undesirable, as improvement in calibration should not come at a price of noticeable drop in predictive correctness. As a counter example, one can achieve perfect calibration in terms of zero ECE by predicting marginal class probability, but this prediction is useless in practice. For image classification on CIFAR100, KFLLLA achieves competitive ECE compared with SGPA, however, it achieves worse NLL and the worst MCE among all the methods. Overall, SGPA achieves the best performance when compared with the other "single-model" baselines: it consistently achieves better calibration across all tasks while

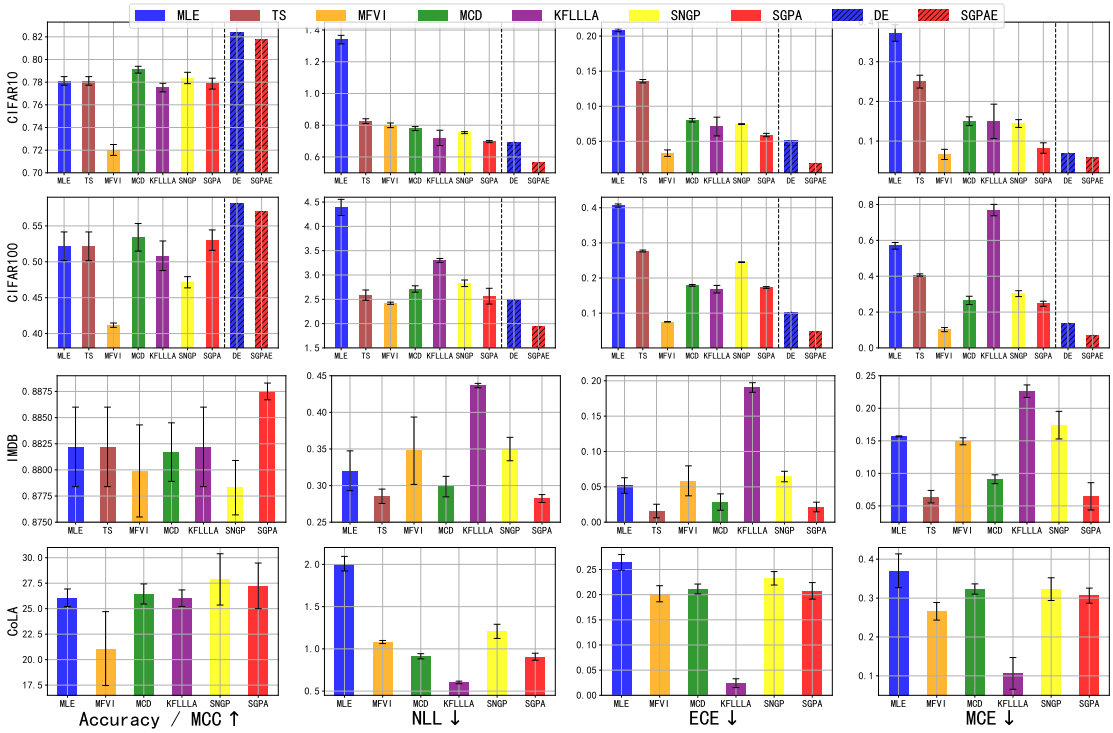


Figure 3.2: Test set accuracy (or MCC for CoLA) & calibration metrics of Transformers or ViTs trained on CIFAR10 (1st row), CIFAR100 (2nd row), IMDB (3rd row) and CoLA (4th row).

maintaining competitive (or even better, on IMDB) predictive accuracy. Compared with "single-model" methods, both ensemble methods, DE and SGPAE, achieve much better predictive accuracy. SGPAE noticeably outperforms DE in terms of calibration while maintaining competitive predictive accuracy.

For CIFAR10 and CIFAR100, we also consider training ViTs with data augmentation and report the results of in-distribution calibration in Figure 3.3. Although some "single-model" methods can achieve lower ECE or MCE than DE and SGPAE in some

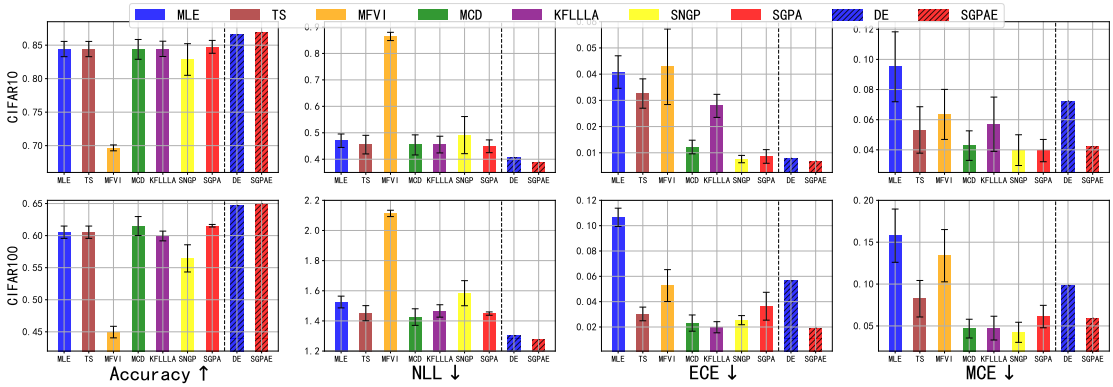


Figure 3.3: Test set accuracy & calibration metrics of ViTs trained on CIFAR10 (top row) and CIFAR100 (bottom row) with data augmentation.

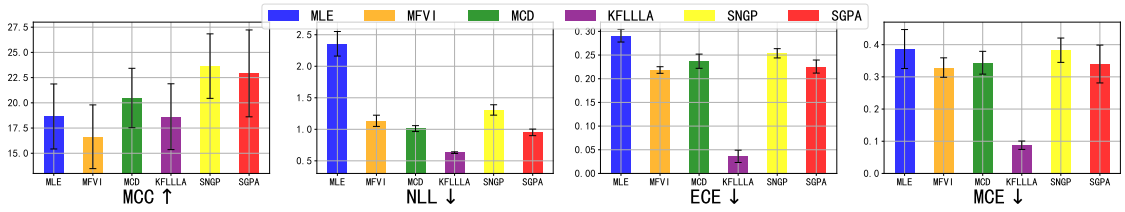


Figure 3.4: Test set MCC & calibration metrics for OOD test set of Transformers trained on COLA.

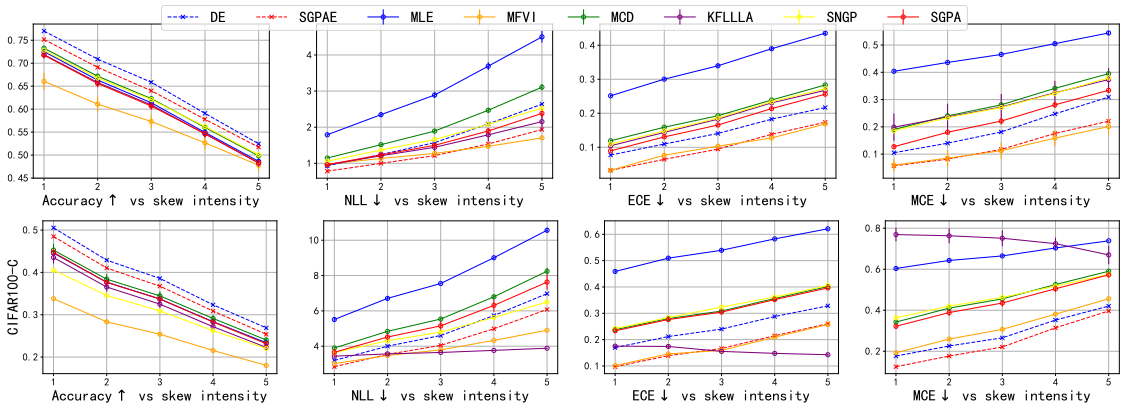


Figure 3.5: Test set accuracy & calibration metrics on CIFAR10-C (top row) and CIFAR100-C (bottom row) against skew intensity of corruption for ViTs trained on corresponding clean data without data augmentation.

cases, DE and SGPAE consistently outperform them in terms of accuracy and NLL. SGPAE again achieves the best overall performance. Among "single-model"methods, MFVI still underfits the data, but for the other methods, data augmentation improves the model performance with SNGP achieving relatively low accuracy. The difference between SGPA and other "single-model" baselines becomes smaller, perhaps due to the strong regularization from data augmentation. Still, SGPA performs more robustly as it generally returns smaller error bars when compared to TS, MCD, and KFLLLA.

3.5.2 Robust Prediction on Out-of-distribution Data

Next we evaluate the performance of SGPA under distribution shift for both the linguistic acceptability task (CoLA) and the image classification tasks (CIFAR10 & CIFAR100). The OOD data for CoLA is introduced by the same authors of Warstadt et al. (2019), while for the CIFAR datasets, we use corrupted CIFAR datasets (CIFAR10-C and CIFAR100-C) (Hendrycks and Dietterich, 2019) as the OOD data, which contains noisy CIFAR images with different types of distortions introduced to their clean counterparts at different skew intensities. Note that we don't consider TS for OOD tasks in this and the next section since as a Frequentist

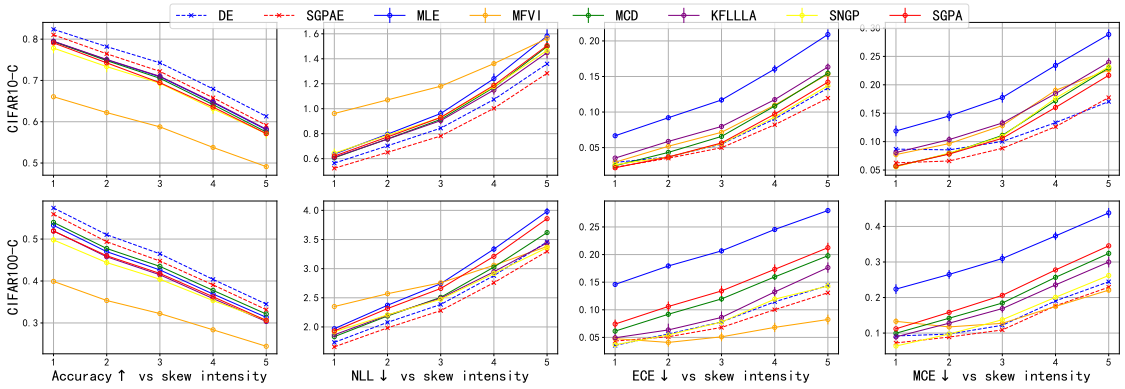


Figure 3.6: Test set accuracy & calibration metrics on CIFAR10-C (top row) and CIFAR100-C (bottom row) against skew intensity of corruption for ViTs trained on corresponding clean data with data augmentation.

method, it is proposed to calibrate the uncertainty on in-distribution data only.

We report in Figure 3.4 the MCC and calibration metrics for the OOD test on CoLA. The observations are similar with the in-distribution test: SGPA outperforms MLE, MCD, and SNGP in terms of NLL and calibration errors while achieving improved accuracy. MFVI and KFLLLA achieve lower calibration errors but they achieve worse predictive accuracy than SGPA. In particular, MFVI again underfits the data.

For OOD robustness test on image classification, we compute metrics against skew intensity on the corrupted CIFAR datasets, and report the results without data augmentation in Figure 3.5 and the results with data augmentation in Figure 3.6. Without data augmentation, among "single-model" methods, SGPA outperforms MLE, MCD and SNGP in terms of calibration without hurting accuracy. MFVI achieves lower calibration errors than SGPA but it pays the price of underfitting especially when the skew intensity is small. The performance of KFLLLA seems to be not as stable as SGPA. For CIFAR10-C, SGPA achieves better calibration than

Table 3.2: Average ranks of different methods in terms of AUROC and AUPR over 6 OOD detection tasks

	Without data augmentation		With data augmentation	
Model	rank (AUROC) ↓	rank (AUPR) ↓	rank (AUROC) ↓	rank (AUPR) ↓
MLE	6.1667	5.5000	6.1667	6.0000
MFVI	7.0000	7.1667	8.0000	8.0000
MCD	5.6667	5.6667	4.1667	4.5000
KFLLLA	4.3333	4.5000	4.3333	4.0000
SNGP	5.3333	5.3333	6.3333	6.3333
SGPA	4.5000	4.6667	4.0000	4.1667
DE	1.6667	2.0000	1.8333	1.8333
SGPAE	**1.3333**	**1.1667**	**1.1667**	**1.1667**

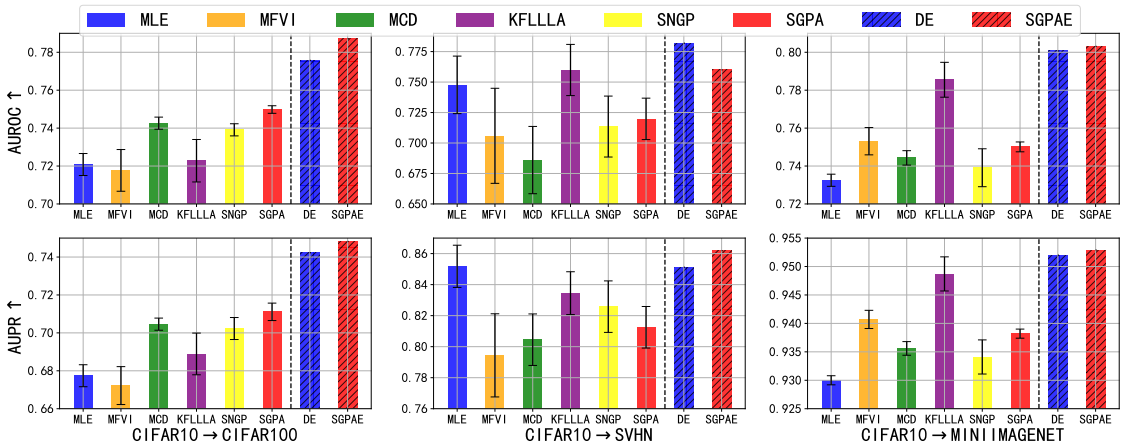


Figure 3.7: AUROC (top) and AUPR (bottom) metrics for OOD detection using ViTs trained on CIFAR10 without data augmentation.

KFLLLA. For CIFAR100-C, KFLLA achieves the best NLL and ECE but the worst MCE. When the data augmentation is applied, MFVI still suffers from underfitting, but the performance gap between SGPA and other "single-model" baselines again becomes smaller. Regardless of using data augmentation or not, ensemble methods achieve better accuracy than "single-model" methods and SGPAE still outperforms DE in terms of calibration while achieving similar accuracy.

3.5.3 Out-of-distribution Detection

Lastly we consider OOD detection tasks on Transformer models trained for image classification, to further evaluate the quality of uncertainty estimates. Here we use predictive entropy to score each input (from either in-distribution or OOD data) and make decisions of "in/out" if the entropy is smaller/greater than a specified threshold. Using different thresholds for this detection task allows us to compute

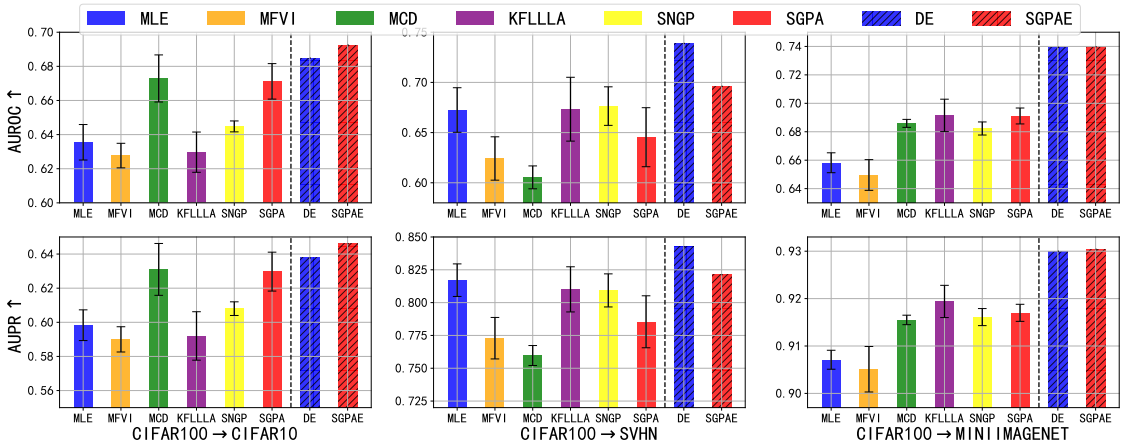


Figure 3.8: AUROC (top) and AUPR (bottom) metrics for OOD detection using ViTs trained on CIFAR100 without data augmentation.

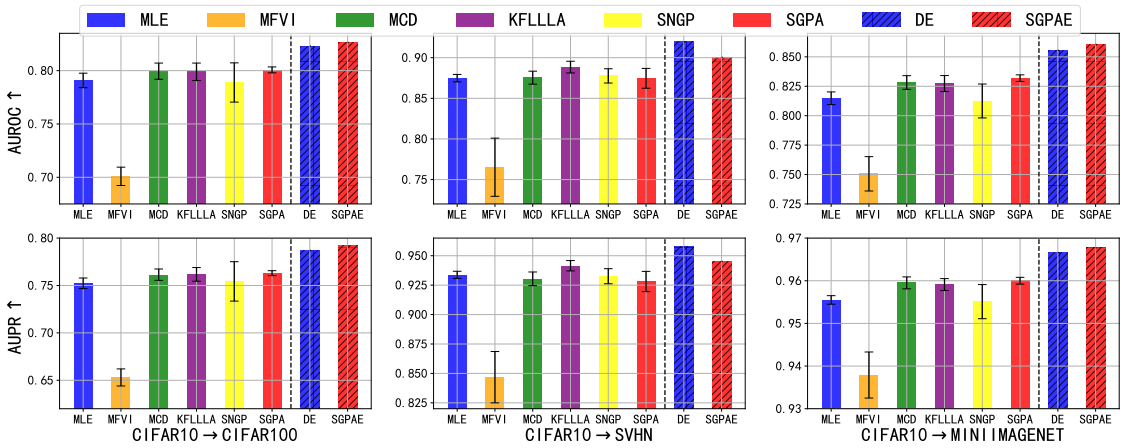


Figure 3.9: AUROC (top) and AUPR (bottom) metrics for OOD detection using ViTs trained on CIFAR10 with data augmentation.

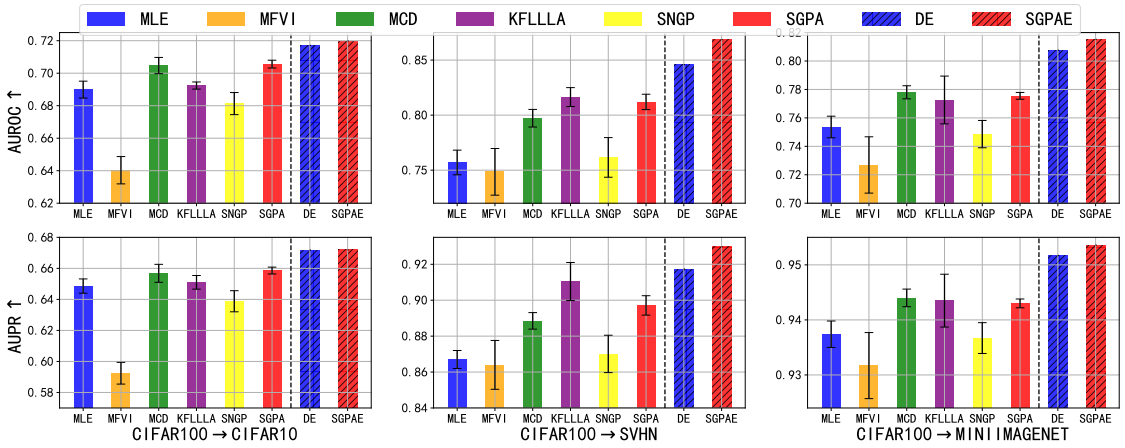


Figure 3.10: AUROC (top) and AUPR (bottom) metrics for OOD detection using ViTs trained on CIFAR100 with data augmentation.

both the receiver operator characteristic (ROC) and the precision-recall (PR) curves, and we use the area under the curves, i.e., AUROC and AUPR, for performance evaluations.

For each of the two CIFAR datasets, we consider the other CIFAR dataset, SVHN and mini-ImageNet as OOD datasets so that we construct 6 OOD detection tasks in total. For each method trained with or without data augmentation, we report its average ranks in terms of AUROC and AUPR over 6 tasks in Table 3.2. Ensemble methods outperform "single-model" methods with SGPAE achieving the best performance. Among "single-model" methods, apart from KFLLLA which achieves the best performance, SGPA outperforms all the other baselines. For comparison within each task, we plot the values of AUROC and AUPR achieved by all methods trained without data augmentation for each task in Figure 3.7 and 3.8, and in Figure 3.9 and 3.10, we plot the values of AUROC and AUPR achieved by all methods trained

with data augmentation within each task.

3.5.4 Summary of Additional Results

Additional experiments are reported in Appendix B.3 where we summarize the main results as follows.

- In Appendix B.3.1, we find that in addition to the parameter inefficiency problem, Transformers based on standard SGPA also suffer from underfitting. Compared with decoupled SGPA, standard SGPA achieves significantly worse accuracy on CIFAR10 classification task.
- In Appendix B.3.2, we report results of an additional experiment, graph property regression with ZINC dataset (Dwivedi et al., 2020). For this task, SGPAE achieves the best performance and SGPA outperforms the other "single-model" baselines.

3.6 Related Work

Bayesian Transformers. Tran et al. (2019) and Xue et al. (2021) propose to perform approximate posterior inference using MFVI in weight space for a subset of layers in Transformers. However, in our experiments we find this type of approaches underfits the data. This pathology of underfitting is theoretically confirmed for weight space MFVI (Foong et al., 2020; Coker et al., 2022). Another line of research proposes to perform VI over the attention matrices directly (Fan et al., 2020; Cinquin et al., 2021). However, Fan et al. (2020) only considers finetuning with variational attention, and (Cinquin et al., 2021) only considers experiments on synthetic or simple datasets with shallow networks and the variational distribution fitted over the attention weights are shared across data, which might be too restrictive for complex problems. Moreover, they find that a data-dependent variational distribution over attention weights can even hurt the performance of their approaches. Liu et al. (2020) consider performing Bayesian inference directly over the Transformer output by fitting a GP over the last layer output (Bradshaw et al., 2017). This approach can be viewed as using a GP model with a deep kernel defined by the Transformer. Instead, SGPA fits a deep GP so that uncertainty is propagated through each attention layer of the Transformer. In addition, Liu et al. (2020) propose to preserve the distance awareness property for the deep kernel. Note that this distance-preserving trick is orthogonal to ours and can also be easily integrated into SGPA.

Related GP methods. The ELBO of SGPA is similar to that in Sun et al. (2021) which also propose to independently approximate the posterior for each additive

component in an additive GP (Duvenaud et al., 2011). The difference is in the kernel design: Sun et al. (2021) aim to decompose a given kernel function into orthogonal "kernel basis", while in SGPA we consider the same type of kernel for each attention head but with different kernel hyperparameters. Our approach is also related to sparse within sparse Gaussian process (SWSGP) (Tran et al., 2021; Jafrasteh et al., 2022) which allows adaptive inducing points for each data point (similar to the input-dependent keys $\boldsymbol{k}_a$ in SGPA). This connection between SGPA and SWSGP is further discussed in Appendix B.6.

3.7 Conclusion and Future Work

We have proposed SGPA to directly perform approximate Bayesian inference over the output of attention blocks in Transformers. Unlike other Bayesian neural network approaches, SGPA takes advantage of the inductive bias embedded in the attention architecture to build Bayesian Transformers. Moreover, we showed Transformers based on SGPA achieve better balance between predictive accuracy and calibration compared with other baselines. Furthermore, the improved quality of uncertainty estimation provided by SGPA has been proved useful in maintaining robustness under distribution shift and in out of distribution detection.

The following directions are interesting to explore as future work. First, masked pre-training (Devlin et al., 2019), which has been proved crucial for downstream tasks for standard Transformers, may also improve the performance of Transformers based on SGPA. In this work, we are not able to consider pretraining due to the high computational cost, and since SGPA replaces scaled dot-product with a valid kernel, there is no existing pre-trained backbone that can be directly used for the downstream fine-tuning tasks. Second, many modern tasks using Transformers require next token prediction in an autoregressive manner, which involve decoder-based architectures (Vaswani et al., 2017; Brown et al., 2020; OpenAI et al., 2024b). While we only consider encoder-based Transformers in this work, extending SGPA to decoder-based Transformers will further broaden the impact of this method, and We give one potential solution in Chapter 6.

Chapter 4

Recurrent Memory for Online Interdomain Gaussian Processes

This chapter is based on Chen et al. (2025b):

- Wenlong Chen*, Naoki Kiyohara*, Harrison Bo Hua Zhu*, Jacob Curran-Sebastian, Samir Bhatt, and Yingzhen Li. **Recurrent Memory for Online Interdomain Gaussian Processes**. In *Advances in Neural Processing Systems (NeurIPS)*, 2025.

The main idea was developed by me while the last author provided useful suggestions. The three co-first authors (*) all contributed to the code implementation, experimentation, and paper writing under the supervision of the last author. Moreover, I performed all the derivations and helped orchestrate other authors' contributions.

As we briefly mentioned in Section 2.2.1, Bayesian inference provides a natural framework for online or continual learning. In this chapter, we focus on improving approximate online Bayesian inference in sparse GP models, again, by exploiting inductive bias in deep sequence models. Specifically, we propose HiPPO-SVGP which leverages the long-range memory preservation capability of HiPPO, a powerful RNN architecture reviewed in Section 2.1.2, to construct a better sparse approximation that mitigates the catastrophic forgetting issue in online sparse GP methods.

4.1 Introduction

Gaussian processes (GPs) are popular choices for modeling time series due to their functional expressiveness and uncertainty quantification abilities (Roberts et al., 2013; Fortuin et al., 2020). However, GPs are computationally expensive and memory

intensive, with cubic and quadratic complexities, respectively. In online regression settings, such as weather modeling, the number of time steps can be very large, quickly making GPs infeasible. Although variational approximations, such as utilizing sparse inducing points (SGPR (Titsias, 2009); SVGP (Hensman et al., 2013, 2015)) and Markovian GPs (Särkkä and Solin, 2019; Wilkinson et al., 2021), have been proposed to address the computational complexity, it would still be prohibitive to re-fit the GP model from scratch every time new data arrives. Bui et al. (2017) proposed an online sparse variational GP (OSVGP) learning method that sequentially updates the GP posterior distribution only based on the newly arrived data. However, as indicated in their paper, their models may not maintain the memory of the previous data, as the inducing locations will inevitably shift as new data arrive. This is a major drawback, as their models may not model long-term memory unless using a growing number of inducing points.

In deep learning, as an alternative to Transformers (Vaswani et al., 2017), significant works on state space models (SSMs) have been proposed to model long-term memory in sequential data. Originally proposed to instill long-term memory in recurrent neural networks, the HiPPO (High-order Polynomial Projection Operators) framework (Gu et al., 2020) provides mathematical foundations for compressing continuous-time signals into memory states through orthogonal polynomial projections. HiPPO is computationally efficient and exhibits strong performance in long-range memory tasks, and forms the basis for the state-of-the-art SSMs, e.g., structured state space sequential (S4) model (Gu et al., 2022) and Mamba (Gu and Dao, 2023; Dao and Gu, 2024).

Inspired by HiPPO, we propose Online HiPPO SVGP (OHSVGP), by applying the HiPPO framework to SVGP in order to leverage the long-range memory modeling capabilities. Our method interprets the HiPPO time-varying orthogonal projections as inducing variables of an interdomain SVGP (Lázaro-Gredilla and Figueiras-Vidal, 2009; Leibfried et al., 2020; Van der Wilk et al., 2020), where the basis functions are time-dependent orthogonal polynomials. We show that we are able to significantly resolve the memory-loss issue in OSVGP, thereby opening up the possibility of applying GPs to long-term online learning tasks. In summary, our contributions include:

- (Section 4.3) We demonstrate that HiPPO integrates into the interdomain GPs by interpreting the HiPPO projections as inducing variables with time-dependent orthogonal polynomial basis functions. This allows the inducing variables to compress historical data, capturing long-term information.

- (Section 4.3.2 & 4.6.1) We show that the kernel matrices can leverage the

efficient ODE evolution of the HiPPO framework, bringing an extra layer of computational efficiency to OHSVGP.

- (Section 4.6) We demonstrate OHSVGP on a variety of online/continual learning tasks including time series prediction, continual learning on UCI benchmarks, and continual learning in Gaussian process variational autoencoder, showing that it outperforms other online sparse GP baselines in terms of predictive performance, long-term memory preservation, and computational efficiency.

4.2 Preliminaries

In this section, we provide a brief overview of interdomain inducing point methods, online learning with sparse GPs, and Gaussian process variational autoencoders.

4.2.1 Interdomain Gaussian Processes

Interdomain GPs are almost the same as the standard SVGP, and the only difference is that instead of setting the inducing variables as the function values evaluated at some input locations $\boldsymbol{Z}$, interdomain GPs (Lázaro-Gredilla and Figueiras-Vidal, 2009; Van der Wilk et al., 2020) propose to generalize the inducing variables to

$$u_m := \int f(x)\phi_m(x)\mathrm{d}x, \tag{4.1}$$

where $\phi_m(x)$ are basis functions, to allow for further flexibility. Since integration is a linear operator and f is a GP, u_m follows a Gaussian distribution[1]. The prior cross-covariance between the function values $f_x := f(x)$ and the inducing variables, and the prior inducing covariance between the inducing variables themselves are computed as integrals rather than direct kernel evaluations:

$$\begin{aligned}[\mathbf{K}_{\mathbf{fu}}]_m &:= \mathrm{COV}(f_x, u_m) = \int k(x, x')\phi_m(x')\mathrm{d}x', \\ [\mathbf{K}_{\mathbf{uu}}]_{nm} &:= \mathrm{COV}(u_n, u_m,) = \iint k(x, x')\phi_n(x)\phi_m(x')\mathrm{d}x\mathrm{d}x'.\end{aligned} \tag{4.2}$$

After replacing the standard prior covariance matrices $\boldsymbol{K_{XZ}}$ and $\boldsymbol{K_{ZZ}}$ with $\boldsymbol{K_{fu}}$ and $\boldsymbol{K_{uu}}$ defined above, the posterior predictive distribution and the ELBO training objective for interdomain GPs remain exactly the same as the standard SVGP shown in Section 2.3.2.

We see that unlike standard SVGP based on the inducing locations $\mathbf{Z} \in \mathbb{R}^M$, which can shift locations according to the training data, the interdomain GP bypasses the

[1] In fact, interdomain u can be defined by applying any linear operator L to the prior GP $f(\cdot)$ such that $u := Lf(\cdot)$. In this work, we focus on the case where L is an integral operator.

selection of the inducing locations, and reformulates it with the selection of the basis functions ϕ_i. The basis functions dictate the structure of the two covariance/kernel matrices above, which in turn modulate the function space of the GP approximation. For example, some interdomian GPs utilize properties of some basis functions to sparsify $\boldsymbol{K_{uu}}$ and therefore reduce the computational cost for its inversion (Hensman et al., 2018; Dutordoir et al., 2020).

4.2.2 Online Sparse Gaussian Processes

Consider online learning where data arrives sequentially in batches $\mathcal{D}_{t_1} := (\mathbf{X}_{t_1}, \mathbf{y}_{t_1})$, $\mathcal{D}_{t_2} := (\mathbf{X}_{t_2}, \mathbf{y}_{t_2}), \ldots$ etc. For example, in the time series prediction setting, the data arrives in intervals of $(0, t_1), (t_1, t_2), \ldots$ etc. Sequential Bayesian inference provides a principled framework for online learning where the old posterior for the model can be used as the new prior for the new task. Combining with the likelihood based on the new data, we can then obtain the new posterior. Again, approximation is required when the true sequential posterior update is intractable.

In this work, we focus on online learning with sparse GPs (reviwed in Section 2.3.2), which sequentially update the sparse GP posterior distribution as new data arrive. Suppose that we have already obtained an SVGP (with inducing points $(\mathbf{Z}_{t_1}, \mathbf{u}_{t_1})$), $q_{t_1}(f) = \int p_{t_1}(f|\mathbf{u}_{t_1}) q_{t_1}(\mathbf{u}_{t_1}) d\mathbf{u}_{t_1}$, trained on the first batch of data, $\mathcal{D}_{t_1}$, online SVGP (OSVGP; (Bui et al., 2017)) utilizes VI based approximation to update the SVGP when observing the second task. The new SVGP, based on an updated set of inducing points $(\mathbf{Z}_{t_2}, \mathbf{u}_{t_2})$, is denoted as $q_{t_2}(f) = \int p_{t_2}(f|\mathbf{u}_{t_2}) q_{t_2}(\mathbf{u}_{t_2}) d\mathbf{u}_{t_2}$. Notice that the prior for the new SVGP can be updated as well (i.e., the kernel hyperparameters can be updated), so $p_{t_2}(f)$ can be different from $p_{t_1}(f)$. Ultimately, we would like to minimize the full-batch KL-divergence $\mathrm{KL}\left(q_{t_2}(f)||p_{t_2}(f|\mathcal{D}_{t_1}, \mathcal{D}_{t_2})\right)$, which is equivalent to maximizing the following ELBO:

$$\begin{aligned}
\mathcal{L}_{ELBO} &= \log \frac{p_{t_2}(\mathcal{D}_{t_1}, \mathcal{D}_{t_2})}{p_{t_1}(\mathcal{D}_{t_1})} - \mathrm{KL}\left(q_{t_2}(f)||p_{t_2}(f|\mathcal{D}_{t_1}, \mathcal{D}_{t_2})\right) \\
&= \int q_{t_2}(f) \log \frac{p_{t_2}(\mathbf{y}_{t_2}, \mathbf{y}_{t_1})}{p_{t_1}(\mathbf{y}_{t_1}) q_{t_2}(f)} p_{t_2}(f|\mathbf{y}_{t_1}, \mathbf{y}_{t_2}) df \\
&= \int q_{t_2}(f) \log \frac{p_{t_2}(f) p_{t_2}(\mathbf{y}_{t_2}|f) p_{t_2}(\mathbf{y}_{t_1}|f)}{p_{t_1}(\mathbf{y}_{t_1}) q_{t_2}(f)} df.
\end{aligned} \tag{4.3}$$

Notice that $p_{t_1}(\mathcal{D}_{t_1})$ in the denominator of the first term in the first line is a constant since it only depends on the old kernel hyperparameters. Unfortunately, in an online setting, $p_{t_2}(\mathbf{y}_{t_1}|f)$ is intractable since we are not allowed to revisit the old data with the new model. To bypass this intractability, Bui et al. (2017) proposed to

approximate this term with

$$p_{t_2}(\mathbf{y}_{t_1}|f) \approx p_{t_1}(\mathbf{y}_{t_1}|f) = \frac{p_{t_1}(f|\mathbf{y}_{t_1})p_{t_1}(\mathbf{y}_{t_1})}{p_{t_1}(f)} \approx \frac{q_{t_1}(f)p_{t_1}(\mathbf{y}_{t_1})}{p_{t_1}(f)}. \tag{4.4}$$

Plugging this approximation back to Eq. 4.3 yields the "online ELBO":

$$\mathcal{L}^o_{ELBO} = \int q_{t_2}(f) \log \frac{p_{t_2}(f)p_{t_2}(\mathbf{y}_{t_2}|f)q_{t_1}(f)}{q_{t_2}(f)p_{t_1}(f)} df. \tag{4.5}$$

Since $q_{t_1}(f)$ and $q_{t_2}(f)$ are completely determined based on finite sets of inducing points ($\mathbf{u}_{t_1}$ and $\mathbf{u}_{t_2}$, respectively) and we assume that the likelihood factorizes across data points, the objective can be further simplified to be:

$$\begin{aligned} \mathcal{L}^o_{ELBO} = &\sum_{i=1}^{n_{t_2}} \mathbb{E}_{q_{t_2}(f_i)} \left[\log p_{t_2}(y_i \mid f_i)\right] + \mathrm{KL}\left(\tilde{q}_{t_2}(\mathbf{u}_{t_1}) \,\|\, p_{t_1}(\mathbf{u}_{t_1})\right) \\ &- \mathrm{KL}\left(\tilde{q}_{t_2}(\mathbf{u}_{t_1}) \,\|\, q_{t_1}(\mathbf{u}_{t_1})\right) - \mathrm{KL}\left(q_{t_2}(\mathbf{u}_{t_2}) \,\|\, p_{t_2}(\mathbf{u}_{t_2})\right), \end{aligned} \tag{4.6}$$

where $y_i \in \mathbf{y}_{t_2}$ for $i = 1, \ldots, n_{t_2}$ and $\tilde{q}_{t_2}(\mathbf{u}_{t_1}) := \int p_{t_2}(\mathbf{u}_{t_1}|\mathbf{u}_{t_2})q_{t_2}(\mathbf{u}_{t_2})\mathrm{d}\mathbf{u}_{t_2}$.

Unfortunately, with more and more tasks, OSVGP may not capture the long-term memory in the data since as new data arrives, it is not guaranteed that the inducing locations after optimization can sufficiently cover all the previous tasks' input domains.

4.2.3 Gaussian Processes Variational Autoencoders

In addition to predictive tasks, we also consider continual learning in Gaussian processes variational autoencoder (GPVAE; (Casale et al., 2018; Fortuin et al., 2020; Ashman et al., 2020; Jazbec et al., 2021; Zhu et al., 2023)), a GP based generative model, and here we give a brief introduction. GPVAEs embed Gaussian processes within a variational autoencoder (VAE; (Kingma and Welling, 2014)) framework. For sparse GPs with inducing variables, Jazbec et al. (2021) introduced the SVGPVAE, which combines the sparse variational GP (SVGP) with the VAE formulation. The likelihood $p(y \mid \varphi_\theta(f))$ is parameterized by a decoder network φ_θ, which takes GP latent draws f as input, together with the variational inducing posterior $q_\theta(\boldsymbol{u} \mid y)$. This posterior, $q_\theta(\boldsymbol{u} \mid \phi(y))$, is parameterized by the encoder network ϕ. Finally, the latent GP f is typically modeled as a multi-output GP with independent components. GPVAEs have been shown to successfully model high-dimensional time series such as weather data and videos (Zhu et al., 2023; Fortuin et al., 2020). In this work, we consider the SVGPVAE model defined in Jazbec et al. (2021) for one set of our experiments, and the detailed specification of the model and training objective can be found in Appendix C.3.

4.3 Interdomain Inducing Point Gaussian Processes with HiPPO

We bridge the HiPPO framework with interdomain Gaussian processes by interpreting HiPPO's state vector defined by time-varying orthogonal projections as interdomain inducing points. This enables adaptive compression of the history of a GP while preserving long-term memory.

4.3.1 HiPPO as Interdomain Inducing Variables

Recall that in an interdomain setting in Section 4.2.1, inducing variables are defined through an integral transform against a set of basis functions. Let $f \sim \mathcal{GP}(0, k)$, and consider time-dependent basis functions

$$\phi_m^{(t)}(x) = g_m^{(t)}(x)\omega^{(t)}(x), \tag{4.7}$$

where $g_m^{(t)}$ are the orthogonal functions of HiPPO and $\omega^{(t)}$ is the associated measure. We define the corresponding interdomain inducing variables as

$$u_m^{(t)} = \int f(x)\phi_m^{(t)}(x)\mathrm{d}x, \tag{4.8}$$

which is not a one-dimensional random variable as in Section 4.2.1. Rather, it is a random functions (i.e. stochastic processes) over time ($u_m^{(t)} := u_m(t)$) due to time-dependent basis functions. These inducing variables adapt in time, capturing long-range historical information in a compact form via HiPPO's principled polynomial projections.

4.3.2 Adapting the Kernel Matrices over Time

When new observations arrive at later times in a streaming scenario, we must adapt both the prior cross-covariance $\boldsymbol{K_{fu}}$ and the prior covariance of the inducing variables $\boldsymbol{K_{uu}}$. In particular, the basis functions in our HiPPO construction evolve with time, so the corresponding kernel quantities also require updates. Below, we describe how to compute and update these matrices at a new time t_2 given their values at time t_1. For clarity, we first discuss $\boldsymbol{K_{fu}}$, then $\boldsymbol{K_{uu}}$.

Prior cross-covariance $\boldsymbol{K_{fu}^{(t)}}$. Recall that for a single input x_n, the prior cross-covariance with the m-th inducing variable is

$$\left[\boldsymbol{K_{fu}^{(t)}}\right]_{nm} = \int k\left(x_n, x\right)\phi_m^{(t)}(x)\mathrm{d}x, \tag{4.9}$$

which is of the same form as a projection coefficient (Eq. 2.7) in the HiPPO framework. Hence, We can compute the temporal evolution of $\boldsymbol{K}_{\boldsymbol{fu}}^{(t)}$ in a manner consistent with the HiPPO approach, leveraging the same parameters $\boldsymbol{A}(t)$ and $\boldsymbol{B}(t)$. Specifically,

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[\boldsymbol{K}_{\boldsymbol{fu}}^{(t)}\right]_{n,:} = \boldsymbol{A}(t)\left[\boldsymbol{K}_{\boldsymbol{fu}}^{(t)}\right]_{n,:} + \boldsymbol{B}(t)k\left(x_n, t\right), \tag{4.10}$$

where $\left[\boldsymbol{K}_{\boldsymbol{fu}}^{(t)}\right]_{n,:}$ is the n-th row of $\boldsymbol{K}_{\boldsymbol{fu}}^{(t)}$. The matrices $\boldsymbol{A}(t)$ and $\boldsymbol{B}(t)$ depend on the specific choice of the HiPPO measure and basis functions. In our experiments, we employ HiPPO-LegS, whose explicit matrix forms are provided in Section 2.1.2. One then discretizes in t (e.g. using an Euler method or a bilinear transform) to obtain a recurrence update rule.

Prior inducing covariance $\boldsymbol{K}_{\boldsymbol{uu}}^{(t)}$. The ℓm-th element of the prior covariance matrix for the inducing variables is given by

$$\left[\boldsymbol{K}_{\boldsymbol{uu}}^{(t)}\right]_{\ell m} = \iint k\left(x, x'\right)\phi_{\ell}^{(t)}(x)\phi_{m}^{(t)}(x')\mathrm{d}x\mathrm{d}x'. \tag{4.11}$$

Since $k(x, x')$ depends on both x and x', a recurrence update rule based on the original HiPPO formulation, which is designed for single integral, can not be obtained directly for $\boldsymbol{K}_{\boldsymbol{uu}}^{(t)}$. Fortunately, for stationary kernels, Bochner Theorem (Rudin, 1994) can be applied to factorize the double integrals into two separate single integrals, which gives rise to Random Fourier Features (RFF) approximation (Rahimi and Recht, 2007): for a stationary kernel $k(x, x') = k(|x - x'|)$, RFF approximates it as follows:

$$\begin{aligned} k(x, x') &= \mathbb{E}_{p(w)}\Big[\cos(wx)\,\cos(wx') + \sin(wx)\,\sin(wx')\Big] \\ &\approx \frac{1}{N}\sum_{n=1}^{N}\left[\cos\left(w_n x\right)\cos\left(w_n x'\right) + \sin\left(w_n x\right)\sin\left(w_n x'\right)\right], \end{aligned} \tag{4.12}$$

where $w_n \sim p(w)$ is the spectral density of the kernel[2]. Substituting this into the double integral (Eq. 4.11) factorizes the dependency on x and x', reducing $[\boldsymbol{K}_{\boldsymbol{uu}}^{(t)}]_{\ell m}$ to addition of products of one-dimensional integrals. Each integral based on a Monte Carlo sample w has the form of either

$$Z_{w,\ell}^{(t)} = \int \cos(wx)\phi_{\ell}^{(t)}(x)\,\mathrm{d}x \quad \text{or} \quad Z_{w,\ell}^{\prime(t)} = \int \sin(wx)\phi_{\ell}^{(t)}(x)\,\mathrm{d}x, \tag{4.13}$$

[2]For non-stationary kernels, more advanced Fourier feature approximation techniques (e.g., (Ton et al., 2018)) can be applied.

which corresponds to a standard projection coefficient in the HiPPO framework (Eq. 2.7). We further stack these integrals based on M basis functions and define

$$\boldsymbol{Z}_w^{(t)} = \left[Z_{w,1}^{(t)}, \cdots, Z_{w,M}^{(t)}\right]^\top, \quad \boldsymbol{Z}_w'^{(t)} = \left[Z_{w,1}'^{(t)}, \cdots, Z_{w,M}'^{(t)}\right]^\top. \tag{4.14}$$

Collecting N Monte Carlo samples $\{w_n\}_{n=1}^N$, we form the feature matrix

$$\boldsymbol{Z}^{(t)} = \left[\boldsymbol{Z}_{w_1}^{(t)} \quad \boldsymbol{Z}_{w_2}^{(t)} \quad \cdots \quad \boldsymbol{Z}_{w_N}^{(t)} \quad \boldsymbol{Z}_{w_1}'^{(t)} \quad \boldsymbol{Z}_{w_2}'^{(t)} \quad \cdots \quad \boldsymbol{Z}_{w_N}'^{(t)}\right], \tag{4.15}$$

and the RFF approximation of the covariance is

$$\boldsymbol{K}_{\boldsymbol{uu}}^{(t)} \approx \frac{1}{N} \boldsymbol{Z}^{(t)} \left(\boldsymbol{Z}^{(t)}\right)^\top. \tag{4.16}$$

Since $\boldsymbol{Z}_{w_n}^{(t)}$ and $\boldsymbol{Z}_{w_n}'^{(t)}$ are standard HiPPO projection coefficient, their computation is governed by the HiPPO ODE evolution as before

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{Z}_{w_n}^{(t)} = \boldsymbol{A}(t)\,\boldsymbol{Z}_{w_n}^{(t)} + \boldsymbol{B}(t)\,h_n(t), \quad \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{Z}_{w_n}'^{(t)} = \boldsymbol{A}(t)\,\boldsymbol{Z}_{w_n}'^{(t)} + \boldsymbol{B}(t)\,h_n'(t), \tag{4.17}$$

with $h_n(t) = \cos(w_n t)$ and $h_n'(t) = \sin(w_n t)$ and these ODEs can be solved in parallel across different Monte Carlo samples. In summary, the procedure involves sampling multiple random features, updating them recurrently to time t, and averaging across samples to obtain RFF approximation of $\boldsymbol{K}_{\boldsymbol{uu}}^{(t)}$.

Alternatively, one may differentiate $\boldsymbol{K}_{\boldsymbol{uu}}^{(t)}$ directly w.r.t. t. This yields a matrix ODE of the form different from the original HiPPO formulation. For details, see Appendix C.1. Empirically, a vanilla implementation of this approach shows numerical unstability (see Appendix C.1). Hence, we conduct our experiments based on the RFF approximation described above.

Sequential variational updates. Having obtained $\boldsymbol{K}_{\boldsymbol{fu}}^{(t_2)}, \boldsymbol{K}_{\boldsymbol{uu}}^{(t_2)}$ at a new time $t_2 > t_1$, we perform variational updates following the online GP framework described in Section 4.2.2. This ensures the posterior at time t_2 remains consistent with both the new data and the previous posterior at time t_1, based on $\boldsymbol{K}_{\boldsymbol{fu}}^{(t_1)}, \boldsymbol{K}_{\boldsymbol{uu}}^{(t_1)}$. Overall, this procedure endows interdomain HiPPO-based GPs with the ability to capture long-term memory online. By viewing the induced kernel transforms as ODEs in time, we efficiently preserve the memory of past observations while adapting our variational posterior in an online fashion. Figure 4.1b illustrates the evolution of the optimal posterior $q(\boldsymbol{u}^{(x)})$ as time x increases on a toy online time series regression problem with two tasks, where x determines the end of the recurrent update for the prior cross and inducing covariance matrices (evolved up to $\boldsymbol{K}_{\boldsymbol{fu}}^{(x)}$ and $\boldsymbol{K}_{\boldsymbol{uu}}^{(x)}$, respectively). Furthermore, when $x > t_1$, we will update $q(\boldsymbol{u}^{(x)})$ online with the

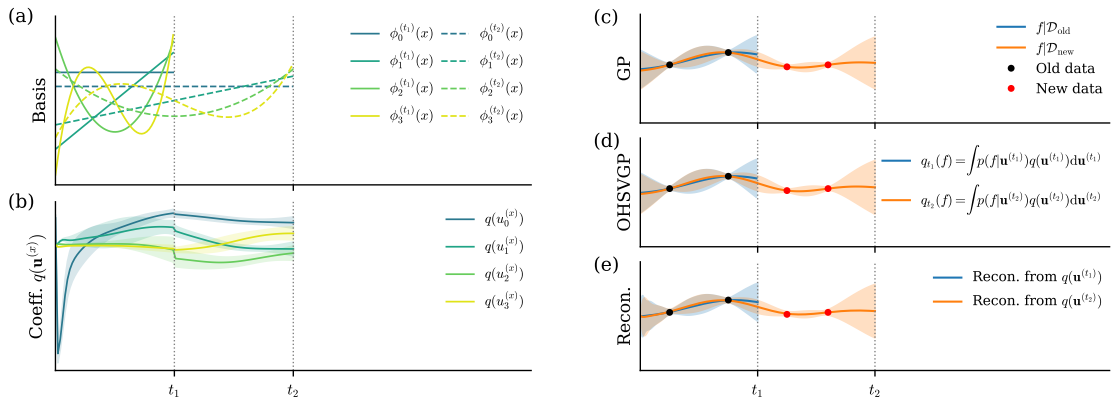


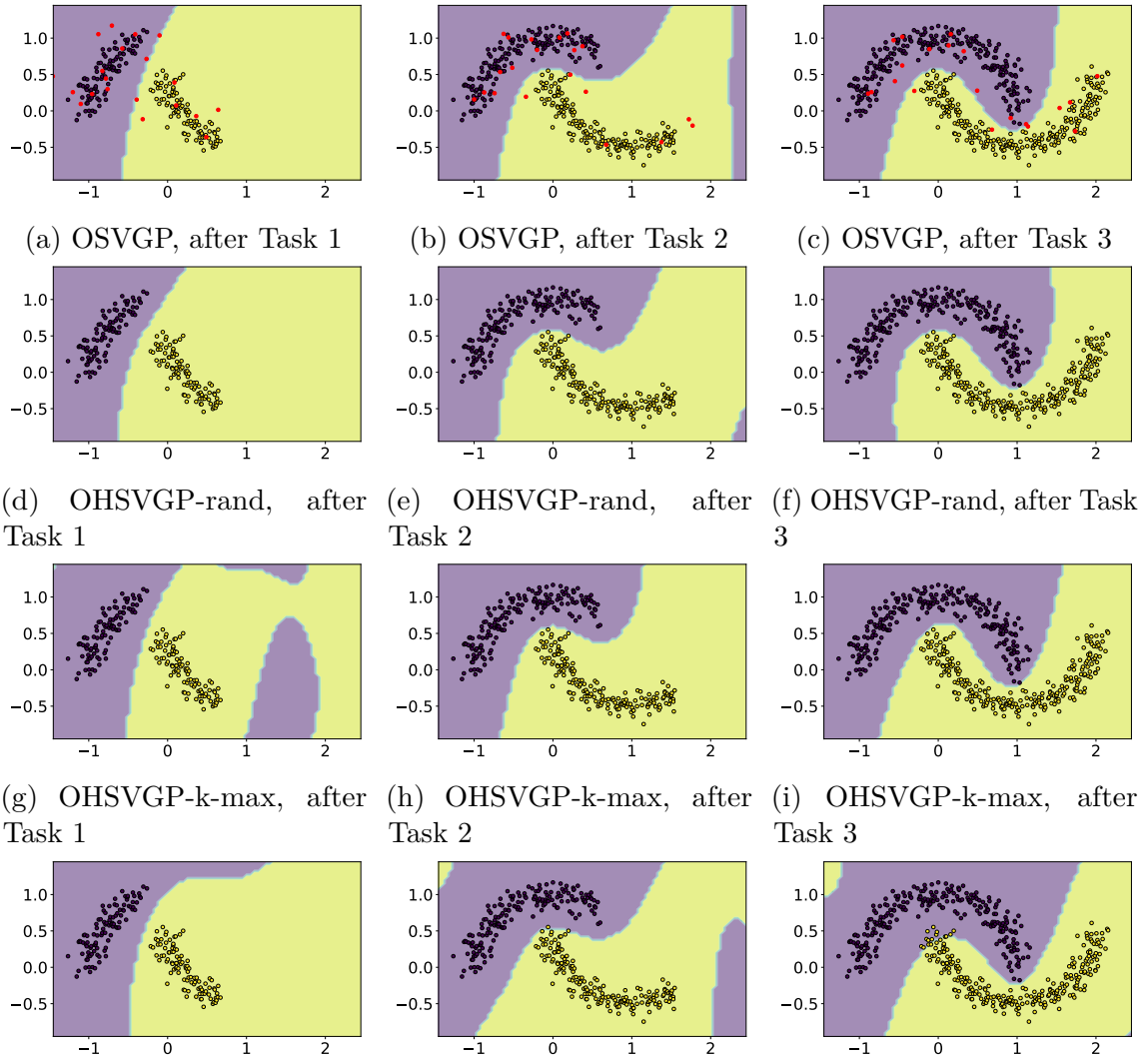
Figure 4.1: Online HiPPO Sparse Variational Gaussian Process (OHSVGP) on a toy time series with 2 tasks. Here x is used to denote arbitrary time index. **(a)** Time-dependent basis functions with end time index $x = t_1$ and $x = t_2$. **(b)** Evolution of optimal approximate posterior of inducing variables (mean ± 2 marginal standard deviation). **(c)**, **(d)**, **(e)** illustrate predictive mean ± 2 standard deviation of posterior online GP, OHSVGP and finite basis reconstruction of posterior OHSVGP, respectively.

two data points from the second task by optimizing the online ELBO (Eq. 4.6), which gives the discrete jump at $x = t_1$. Figure 4.1d shows the posterior OHSVGP compared with the fit of the gold-standard online GP in Figure 4.1c. Notably, if $f \sim q_t(f)$, then $\int f(x)\phi_m^{(t)}(x)\mathrm{d}x \sim q(u_m^{(t)})$ (detailed derivation in Appendix C.2). Therefore, our framework also provides a finite basis approximation of the posterior OHSVGP as a byproduct: $f = \sum_{m=1}^{M} u_m^{(t)} g_m^{(t)}(x)$, $u_m^{(t)} \sim q(u_m^{(t)})$. Figure 4.1e plots the finite basis approximation/reconstruction and it is close to the posterior OHSVGP for this simple example.

4.4 Extending OHSVGP to Multidimensional Input

For multidimensional input data, suppose there is a time order for the first batch of training points with inputs $\{\boldsymbol{x}_n^{(1)}\}_{n=1}^{N_1}$, such that $\boldsymbol{x}_i^{(1)}$ appears after $\boldsymbol{x}_j^{(1)}$ if $i > j$, and we further assume $\boldsymbol{x}_i^{(1)}$ appears at time index $i\Delta t$ (i.e., $\boldsymbol{x}(i\Delta t) = \boldsymbol{x}_i^{(1)}$), where Δt is a user-specified constant step size. In this case, we can again obtain interdomain prior covariance matrices via HiPPO recurrence. For example, a forward Euler method applied to the ODE in Eq. 4.10 for $\boldsymbol{K}_{\boldsymbol{fu}}^t$ yields

$$[\boldsymbol{K}_{\boldsymbol{fu}}^{((i+1)\Delta t)}]_{n,:} = [\boldsymbol{I} + \Delta t \boldsymbol{A}(i\Delta t)][\boldsymbol{K}_{\boldsymbol{fu}}^{(i\Delta t)}]_{n,:} + \Delta t \boldsymbol{B}(i\Delta t) k\left(\boldsymbol{x}_n^{(1)}, \boldsymbol{x}_i^{(1)}\right). \tag{4.18}$$



(a) OSVGP, after Task 1 (b) OSVGP, after Task 2 (c) OSVGP, after Task 3

(d) OHSVGP-rand, after Task 1 (e) OHSVGP-rand, after Task 2 (f) OHSVGP-rand, after Task 3

(g) OHSVGP-k-max, after Task 1 (h) OHSVGP-k-max, after Task 2 (i) OHSVGP-k-max, after Task 3

(j) OHSVGP-k-min, after Task 1 (k) OHSVGP-k-min, after Task 2 (l) OHSVGP-k-min, after Task 3

Figure 4.2: Decision boundaries of OSVGP, and OHSVGP models with different sorting criteria after each task (3 in total) on the Two-moon dataset. For OSVGP, we visualize the inducing locations with red color.

The equation above can be viewed as a discretization (with step size Δt) of an ODE solving path integrals of the form $\int_0^{N_1\Delta t} k\left(\boldsymbol{x}_n^{(1)}, \boldsymbol{x}(s)\right) \phi_m^{(t)}(s) ds$. The i-th training input $\boldsymbol{x}_i^{(1)}$ is assumed to be $\boldsymbol{x}_i^{(1)} := \boldsymbol{x}(i\Delta t)$ and thus the path integral is approximately solved with discretized recurrence based on the training inputs corresponding to $\{\boldsymbol{x}(i\Delta t)\}_{i=1}^{N}$. We continue the recurrence for the second task with ordered training inputs $\{\boldsymbol{x}_n^{(2)}\}_{n=1}^{N_2}$ by assigning time index $(N_1 + i)\Delta t$ to its i-th instance. and keep the recurrence until we learn all the tasks continually. In practice, one may use a multiple of Δt as the step size to accelerate the recurrence, e.g., instead of using all the training inputs, one can compute the recurrence based on $\{\boldsymbol{x}_1, \boldsymbol{x}_3, \boldsymbol{x}_5, \cdots\}$ only by using step size $2\Delta t$.

When there is no natural time order for training instances in each task, such as in standard continual learning applications, we need to sort the instances with some criterion to create pseudo time order to fit OHSVGP, similar to the practice of applying SSMs to non-sequence data modalities, e.g., SSMs, when applied to vision tasks, assign order to patches in an image for recurrence update of the memory (Zhu et al., 2024). In our experiments, we show that the performance of OHSVGP, when applied to continual learning, depends on the sorting criterion used. As an illustrative example, we visualize how different sorting methods impact OHSVGP's performance in continual learning with a 2D continual classification problem. We consider fitting OHSVGPs with 20 inducing variables for a continual binary classification problem on the Two-moon dataset (Ganin et al., 2016). The data is splitted into three tasks and we use a Bernoulli likelihood to model binary labels. We consider three different sorting criteria:

- **Random,** denoted as OHSVGP-rand. The order of data points in each task is obtained via random permutation.

- **Kernel similarity maximization,** denoted as OHSVGP-k-max. We select the i-th point in task j to be $\boldsymbol{x}_i^{(j)} = \arg\max_{\boldsymbol{x}\in\boldsymbol{X}^{(j)}} k(\boldsymbol{x}, \boldsymbol{x}_{i-1}^{(j)})$ for $i > 1$, and the first point in first task is set to be $\boldsymbol{x}_1^{(1)} = \arg\max_{\boldsymbol{x}\in\boldsymbol{X}^{(1)}} k(\boldsymbol{x}, \boldsymbol{0})$. The intuition is that the signals to memorize, when computing the prior covariance matrices, tend to be more smooth if the consecutive $\boldsymbol{x}$'s are close to each other.

- **Kernel similarity minimization,** denoted as OHSVGP-k-min. We select the i-th point in task j to be $\boldsymbol{x}_i^{(j)} = \arg\min_{\boldsymbol{x}\in\boldsymbol{X}^{(j)}} k(\boldsymbol{x}, \boldsymbol{x}_{i-1}^{(j)})$ for $i > 1$, and the first point in first task is set to be $\boldsymbol{x}_1^{(1)} = \arg\min_{\boldsymbol{x}\in\boldsymbol{X}^{(1)}} k(\boldsymbol{x}, \boldsymbol{0})$. In this case, we deliberately make it difficult to memorize the signals in the recurrent computation for the prior covariance matrices.

Figure 4.2 show the decision boundaries after each task for different OHSVGPs

based on different sorting criteria. We also include the decision boundaries of an OSVGP model for reference. Both OHSVGP-k-max and OHSVGP-rand return decision boundaries achieving 100% accuracy, while OHSVGP-k-min show catastrophic forgetting after Task 3, which suggests OHSVGP requires a sensible sorting criterion to perform well in continual learning tasks.

4.5 Related Work

Online sparse GPs. Previous works mainly focus on reducing the sparse approximation error with different approximate inference techniques, such as variational inference (Bui et al., 2017; Maddox et al., 2021), expectation propagation (Csató and Opper, 2002; Bui et al., 2017), Laplace approximation (Maddox et al., 2021), and approximation enhanced with replay buffer (Chang et al., 2023). The orthogonal research problem of online update of inducing points remains relatively underexplored, and pivoted-Cholesky (Burt et al., 2019) as deployed in Maddox et al. (2021); Chang et al. (2023) is one of the most effective approaches for online update of inducing points up to date. We tackle this problem by taking advantage of the long-term memory capability of HiPPO to design an interdomain inducing variable based method and the associated recurrence based online update rules. Notably, our HiPPO inducing variables in principle are compatible with all the aforementioned approximate inference frameworks since only the way of computing prior covariance matrices will be different from standard online sparse GPs.

Interdomain GPs. To our knowledge, OHSVGP is the first interdomain GP method in the context of online learning. Previous interdomain GPs typically construct inducing variables via integration based on a predefined measure (e.g., a uniform measure over a fixed interval (Hensman et al., 2018) or a fixed Gaussian measure (Lázaro-Gredilla and Figueiras-Vidal, 2009)) to prevent diverging covariances, and this predefined measure may not cover all regions where the time indices from future tasks are, making them unsuitable for online learning. In contrast, OHSVGP bypasses this limitation by utilizing adaptive basis functions constructed based on time-dependent measure which keeps extending to the new time region as more tasks arrive.

Markovian GPs. Markovian GPs (Särkkä and Solin, 2019; Wilkinson et al., 2021) have similar recurrence structure during inference and training due to their state space SDE representation. However, Markovian GPs are tailored to one-dimensional input tasks such as time series modeling, while we extend OHSVGP to multidimensional input tasks.

4.6 Experiments

Applications & datasets. We evaluate OHSVGP against baselines in the following tasks.

- **Time series prediction.** We consider regression benchmarks, Solar Irradiance (Lean, 2004), and Audio Signal (Bui and Turner, 2014) produced from the TIMIT database (Garifolo et al., 1993). We preprocess the two datasets following similar procedures described in Gal and Turner (2015) and Bui et al. (2017), respectively (the train-test split is different due to random splitting). In addition, we consider a daily death-count time series from Santa Catarina State, Southern Brazil spanning the March 2020 to February 2021 COVID-19 pandemic, obtained from Hawryluk et al. (2021). We construct online learning tasks by splitting each dataset into 10 (5 for COVID) sequential partitions with an equal number of training instances.

- **Continual learning.** We consider continual learning on two UCI datasets with multi-dim inputs, Skillcraft (Blair et al., 2013) and Powerplant (Tfekci and Kaya, 2014), using the same data preprocessing procedure as in Stanton et al. (2021). We construct two types of continual learning problems by first sorting the data points based on either the values in their first dimension or their L2 distance from the origin, and then splitting the sorted datasets into 10 sequential tasks with an equal number of training instances.

- **High dimensional time series prediction.** We evaluate GPVAEs on hourly climate data from ERA5 (Hersbach et al., 2023), comprising 17 variables across randomly scattered locations around the UK from January 2020 onward. The dataset is split into 10 sequential tasks of 186 hourly time steps each.

Baseline. We compare OHSVGP with OSVGP (Bui et al., 2017) and OVC (Online Variational Conditioning; (Maddox et al., 2021)). At the beginning of each task, OSVGP initialize the inducing locations by sampling from the old inducing locations and the new data inputs, while OVC initializes them via pivoted-Cholesky (Burt et al., 2019) and we consider both fixing the initialized inducing locations as in Chang et al. (2023) (OVC) or keep training them as in Maddox et al. (2021) (OVC-optZ). For time series regression with Gaussian likelihood, we consider OHSGPR and OSGPR (OHSVGP and OSVGP based on closed form ELBO), and we further consider OVFF (OSGPR based on variational Fourier feature (VFF), an interdomain inducing point approach from Hensman et al. (2018)).

Hyperparameters. Within each set of experiments, all the models are trained using Adam (Kingma and Ba, 2015) with the same learning rate and number of iterations. For OHSVGP, we construct inducing variables based on HiPPO-LegS (Gu et al., 2020) (see Appendix C.4.2 for visualizations of using other HiPPO variants) and use 1000 RFF samples. We use ARD-RBF kernel, except for OVFF, tailored specifically to Matérn kernels, where we use Matérn-$\frac{5}{2}$ kernel instead. Similar to Maddox et al. (2021), we do not observe performance gain by keeping updating kernel hyperparameters online, and we include results with trainable kernel hyperparameters in Appendix C.4.1 for time series regression, but the performance becomes unstable when number of tasks is large. Thus, we either only train the kernel hyperparameters during the initial task and keep them fixed thereafter (Section 4.6.3) or obtain them from a full GP model trained over the initial task. It is also worth noting that OVFF requires computing covariances as integrals over a predefined interval covering the whole range of the time indices from all tasks (including unobserved ones), which is impractical in real online learning scenarios. For our experiments, we set the two edges of this interval to be the minimum and maximum time index among the data points from all the tasks, respectively.

Evaluations & metrics. We report results in test NLL (reviewed in Appendix B.2), or commonly referred to as Negative Log Predictive Density (NLPD) in GP literature, and Root Mean Squared Error (RMSE) (Expected Calibration Error (ECE; (Naeini et al., 2015; Guo et al., 2017)) for COVID data instead).

$$\text{NLPD} = -\frac{1}{N}\sum_{i=1}^{N}\log \hat{p}\big(y_i \mid x_i\big), \quad \text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\big(y_i - \hat{y}_i\big)^2}. \tag{4.19}$$

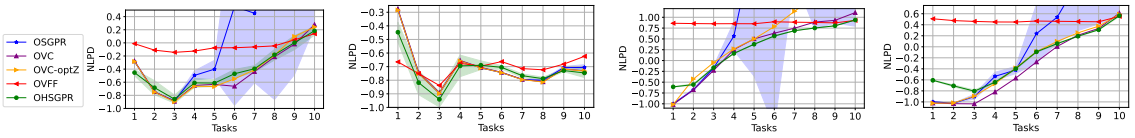
The Expected Calibration Error ((Naeini et al., 2015; Guo et al., 2017); ECE) is defined as the mismatch between the confidence and the coverage:

$$\text{ECE} = \frac{1}{K}\sum_{k=1}^{K}\left|\frac{1}{N}\sum_{i=1}^{N}\mathbf{1}\Big(y_i \in \Big[\hat{q}_{\frac{1-c_k}{2}}(x_i)\ ,\ \hat{q}_{\frac{1+c_k}{2}}(x_i)\Big]\Big) \ - \ c_k,\right| \tag{4.20}$$

where $K = 10, \quad c_k \in \{0.05, 0.15, \ldots, 0.95\}$, N is the number of test points (x_i, y_i), and $\mathbf{1}(\cdot)$ is the indicator function.

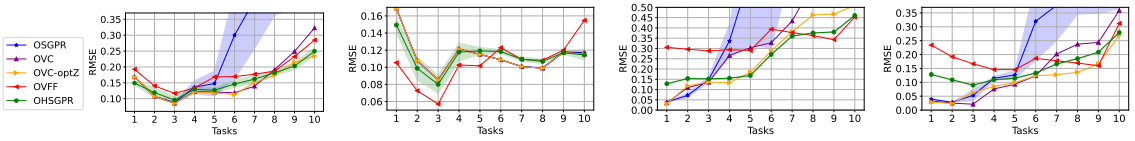
$\hat{q}_p(x_i)$ = the empirical p–quantile of the S predictive samples $\{\hat{y}_i^{(s)}\}_{s=1}^{S}, \quad S = 100.$

We report the mean and the associated 95% confidence interval obtained from 5 (3 for experiments on ERA5) independent runs.



(a) Solar, M=50 (b) Solar, M=150 (c) Audio, M=100 (d) Audio, M=200

Figure 4.3: Test set NLPD over the learned tasks vs. number of learned tasks for Solar Irradiance and Audio signal prediction dataset.

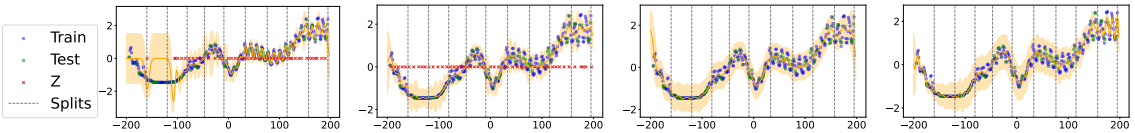


(a) Solar, M=50 (b) Solar, M=150 (c) Audio, M=100 (d) Audio, M=200

Figure 4.4: Test set RMSE over the learned tasks vs. number of learned tasks for Solar Irradiance and Audio signal prediction dataset.

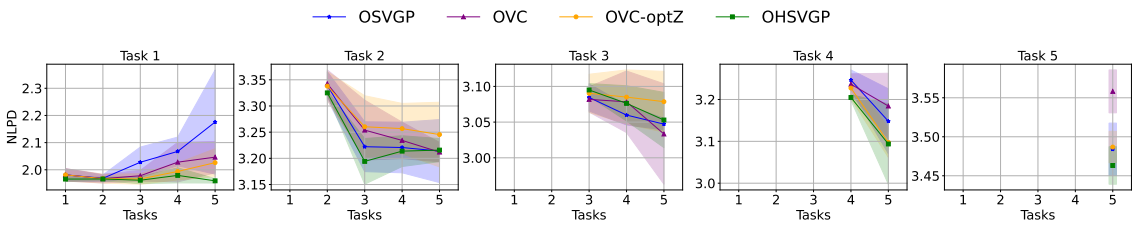
4.6.1 Online Time Series Prediction

Time series regression. Figure 4.3 and Figure 4.4 show NLPD and RMSE (over the past tasks) of different methods during online learning through the 10 tasks for Solar Irradiance and Audio dataset, respectively. Overall, OHSGPR consistently achieves the best performance with OVC performing competitively, especially as we learn more and more tasks, suggesting OHSGPR effectively preserves long-term memory through its HiPPO-based memory mechanism. OSGPR shows catastrophic forgetting starting around task 5, especially when the number of inducing points M is small. Although OVC-optZ also initializes inducing locations with pivoted-Cholesky as OVC, with further optimization, its performance starts to degrade starting from task 6 for the audio dataset when $M = 100$, which suggests the online ELBO objective cannot guarantee optimal online update of inducing locations that preserve memory. OVFF tends to perform well at the later stage. However, during the first few tasks, it underfits the data significantly compared with other methods since its inducing variables are computed via integration over a predefined interval capturing the region of all the tasks, which is unnecessarily long and suboptimal for learning at the early stage.

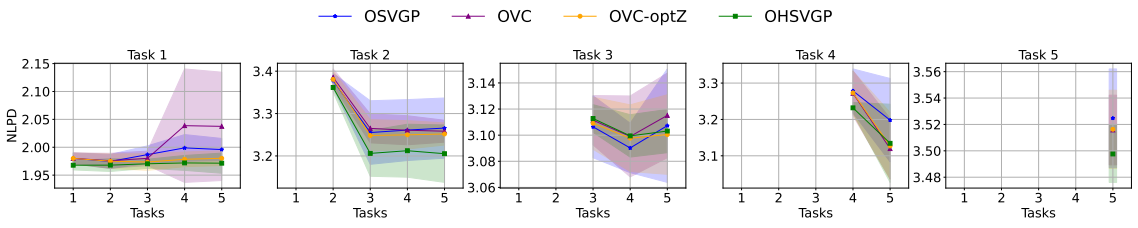


(a) OSGPR (b) OVC (c) OVFF (d) OHSGPR

Figure 4.5: Predictive mean ±2 standard deviation of OSGPR, OVC, OVFF, and OHSGPR after task 10 of the Solar dataset. M = 50 inducing variables are used.



(a) M=15



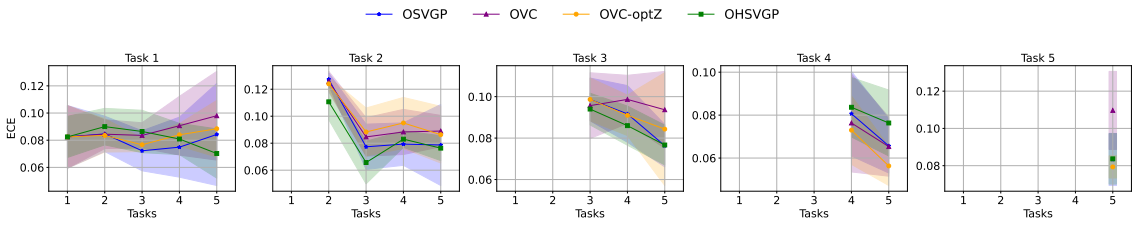
(b) M=30

Figure 4.6: Test set NLPD per task after continually learning each task for all the 5 tasks on COVID dataset.

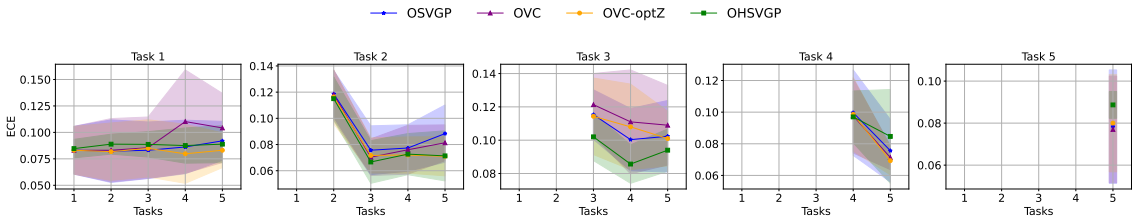
In Figure 4.5, we compare the final predictive distributions for different methods after finishing online learning all 10 tasks of Solar Irradiance. The inducing locations $\boldsymbol{Z}$ for OSGPR tend to move to the regions where the later tasks live after online training, and the prediction of OSGPR in the initial regions without sufficient inducing points becomes close to the uninformative prior GP. In contrast, OHSGPR maintains consistent performance across both early and recent time periods.

Infectious disease modeling We replace the Gaussian likelihood with a Negative Binomial likelihood (non-conjugate) to capture the over-dispersion in COVID-19 death counts. All methods use $M \in \{15, 30\}$ inducing points and are trained for 5000 iterations per task with a learning rate of 0.01. Figure 4.6 and Figure 4.7 show the change of NLPD and ECE through online learning for all five tasks, respectively (i.e., evaluation of Task i after learning tasks $j = i, i+1, \cdots, 5$ for all i). The wide metric variance reflects the noisy nature of death-count data as it is difficult to accurately track down COVID-19 death counts. OHSVGP achieves the best performance overall while OSVGP tend to forget Task 1 with small M.

Runtime comparison. Table 4.1 shows the accumulated wall-clock runtime for different methods to learn all the tasks. Unlike OSVGP and OVC-optZ, which must iteratively optimize inducing locations (for which we train e.g., 1000 iterations for time-series regression tasks), OHSVGP, OVFF (both based on interdomain inducing points), and OVC (based on one-time pivoted-Cholesky update of inducing locations for each task) bypass this cumbersome optimization. In particular, OHSVGP recurrently evolves $\boldsymbol{K_{fu}}$ and $\boldsymbol{K_{uu}}$ for each new task. For regression problems where



(a) M=15



(b) M=30

Figure 4.7: Test set ECE per task after continually learning each task for all the 5 tasks on COVID dataset.

Table 4.1: Wall-clock accumulated runtime for learning all the tasks on a single NVIDIA RTX3090 GPU in seconds, of different models for time series prediction experiments.

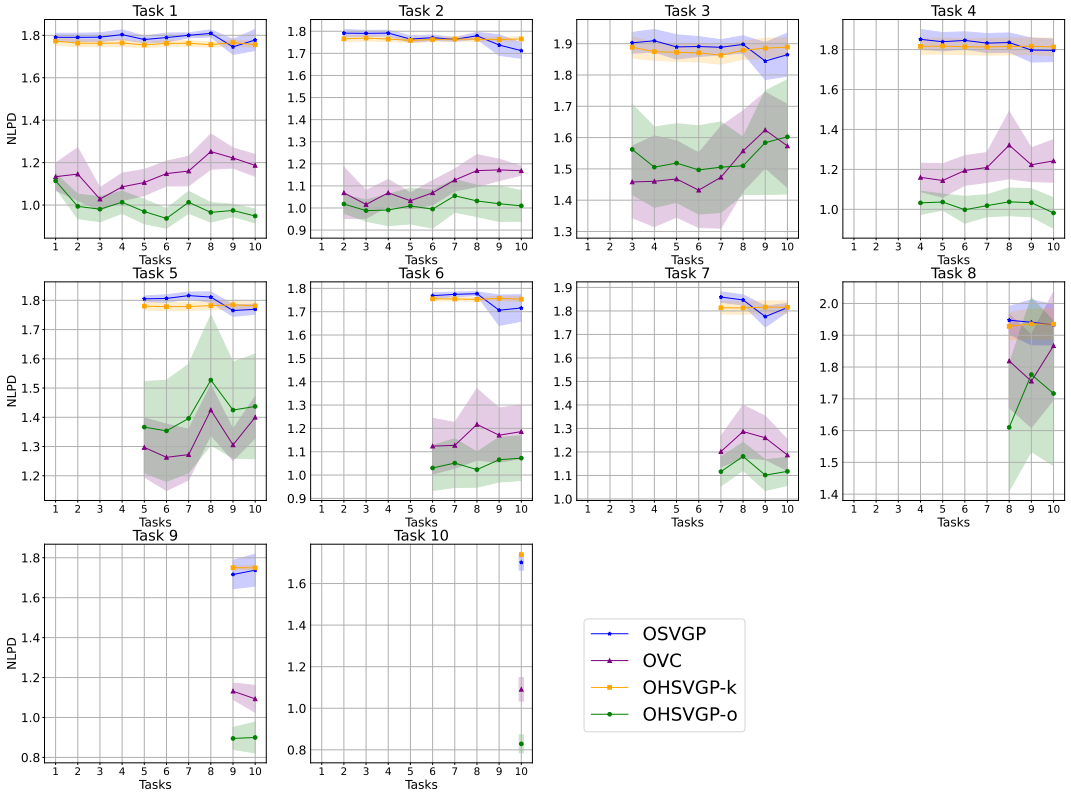
	Solar Irradiance		**Audio Data**		**COVID**	
Method	M		M		M	
	50	150	100	200	15	30
OSGPR/OSVGP	140	149	144	199	525	530
OVC	0.450	0.620	0.558	0.863	345	360
OVFF	0.327	0.354	0.295	0.356	-	-
OHSGPR/OHSVGP	0.297	0.394	0.392	0.655	370	380

closed-form posterior can be obtained, OHSGPR requires no training at all when considering fixed kernel hyperparameters. As a result, OHSGPR, OVC and OVFF run significantly faster, adapting to all tasks within a couple of seconds for Solar Irradiance and Audio data. For COVID data, even when free-form variational parameters of inducing variables are learned using uncollapsed ELBO, OHSVGP and OVC are still significantly faster than OSVGP since no gradient computation is required for the inducing locations.

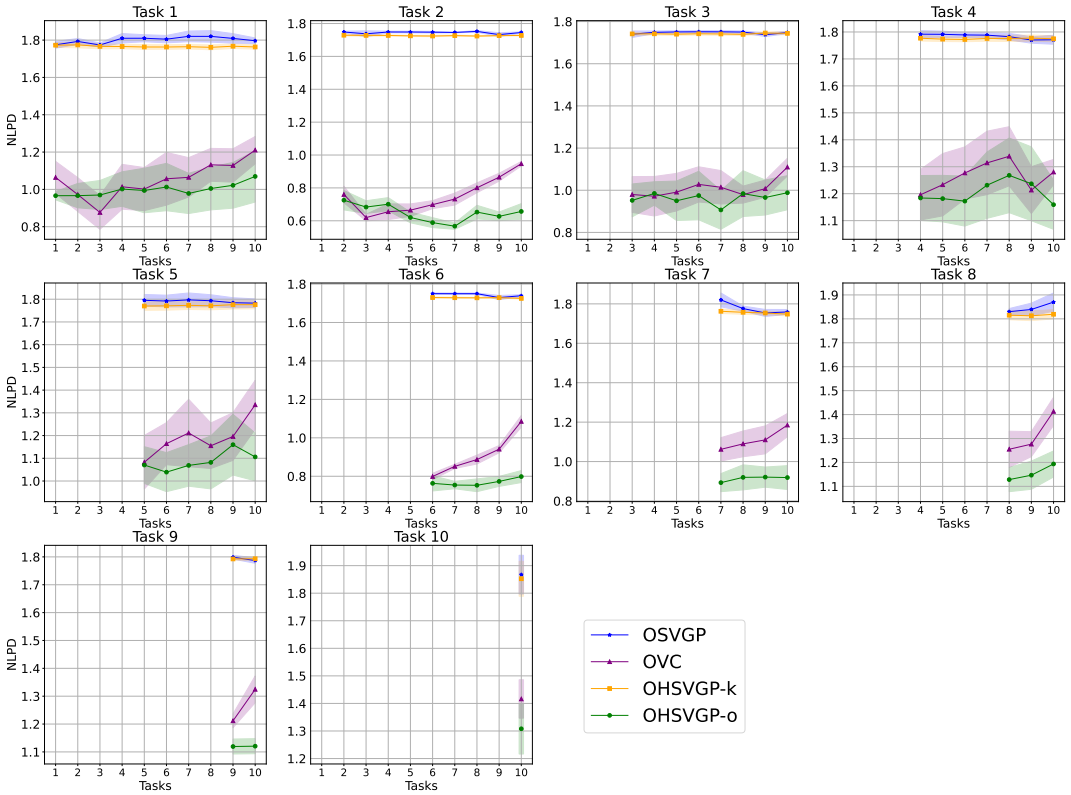
4.6.2 Continual Learning on UCI Datasets

We use 256 inducing variables for all methods, and for each task, we train each method for 2000 iterations with a learning rate of 0.005. We only consider OVC here since initial trials show OVC-optZ give worse results on these two datasets. As described in Section 4.4, within each task, OHSVGP requires sorting the data

points to compute prior covariance matrices via recurrence. We consider two sorting criteria. The first one, which we call OHSVGP-o, uses the oracle order compatible with how the tasks are created (e.g., sort with L2-distance to the origin if the tasks are initially splitted based on it). In real-world problems, we typically do not have the information on how the distribution shifts from task to task. Hence, we also consider OHSVGP-k (same as OHSVGP-k-max in Section 4.4), which uses a heuristic sorting method based on kernel similarity: we select the i-th point in task j to be $\boldsymbol{x}_i^{(j)} = \arg\max_{\boldsymbol{x}\in\boldsymbol{X}^{(j)}} k(\boldsymbol{x}, \boldsymbol{x}_{i-1}^{(j)})$ for $i > 1$, and the first point in first task is set to be $\boldsymbol{x}_1^{(1)} = \arg\max_{\boldsymbol{x}\in\boldsymbol{X}^{(1)}} k(\boldsymbol{x}, \boldsymbol{0})$. Figure 4.8 to Figure 4.11 compares the two variants of OHSVGP with OSVGP and OVC. We report evaluation (in NLPD and RMSE) of Task i after learning tasks $j = i, i+1, \cdots, 10$ for all i. Overall, OSVGP achieves the worst performance and is again prone to forgetting the older tasks, especially in Figure 4.10a. OVC performs decently for Skillcraft but it also demonstrates catastrophic forgetting in Figure 4.10a. While OHSVGP-k achieves similar performance as OSVGP on Skillcraft, OHSVGP-o consistently outperforms the other methods across all 4 scenarios, suggesting the importance of a sensible sorting method when applying OHSVGP for continual learning.

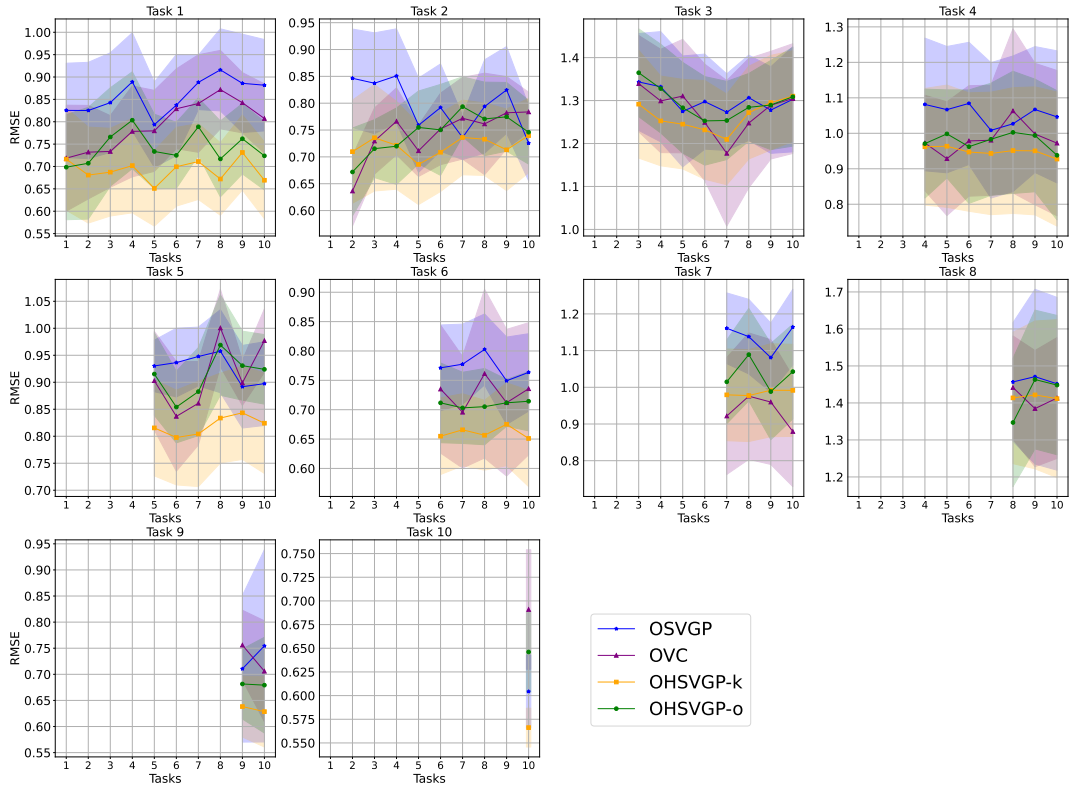


(a) Skillcraft (1st dimension)

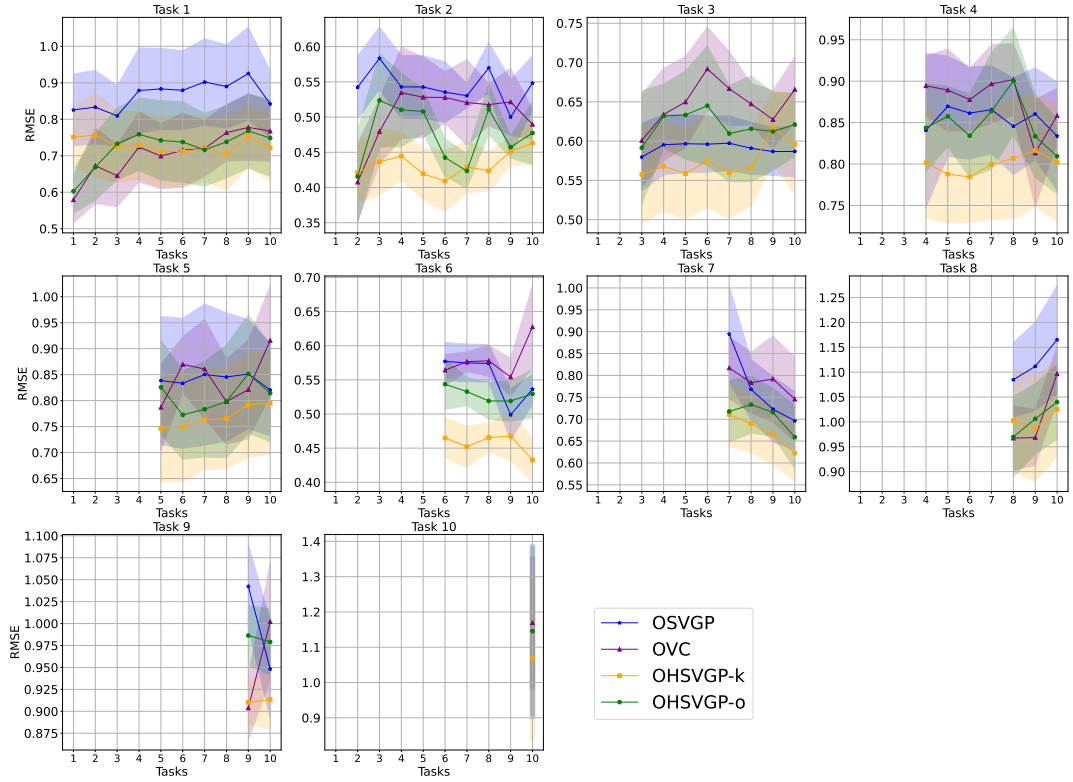


(b) Skillcraft (L2)

Figure 4.8: Test set NLPD per task after continually learning each task for all the 10 tasks on UCI Skillcraft dataset.

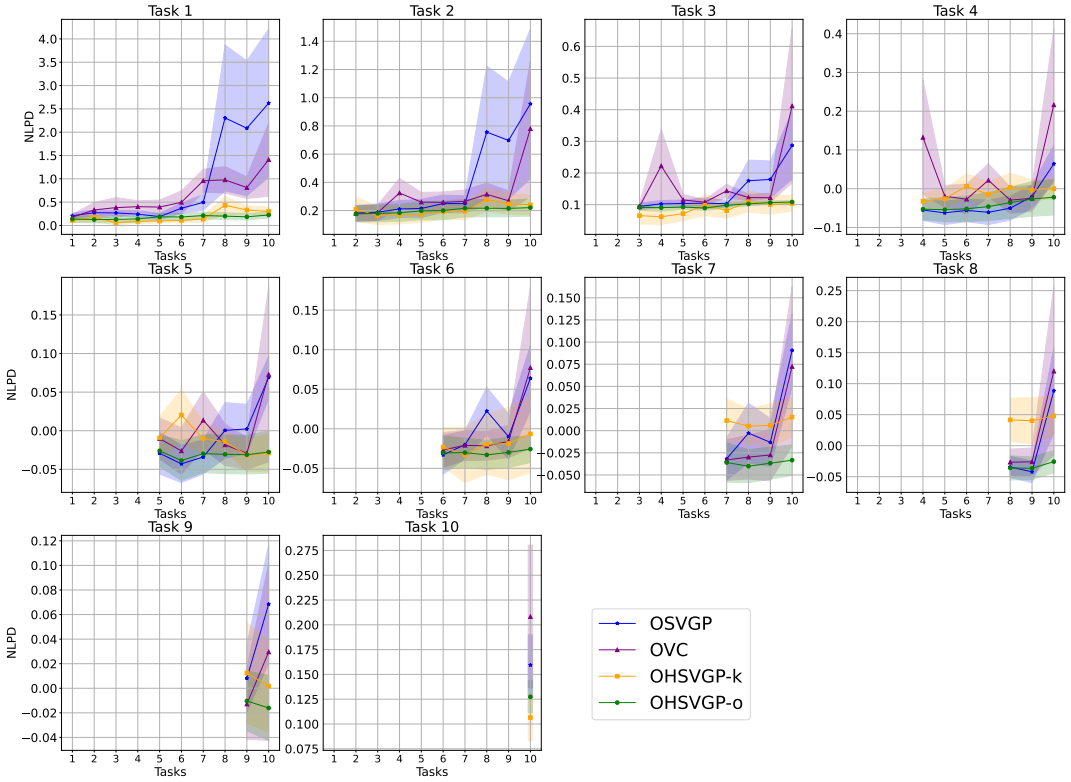


(a) Skillcraft (1st dimension)

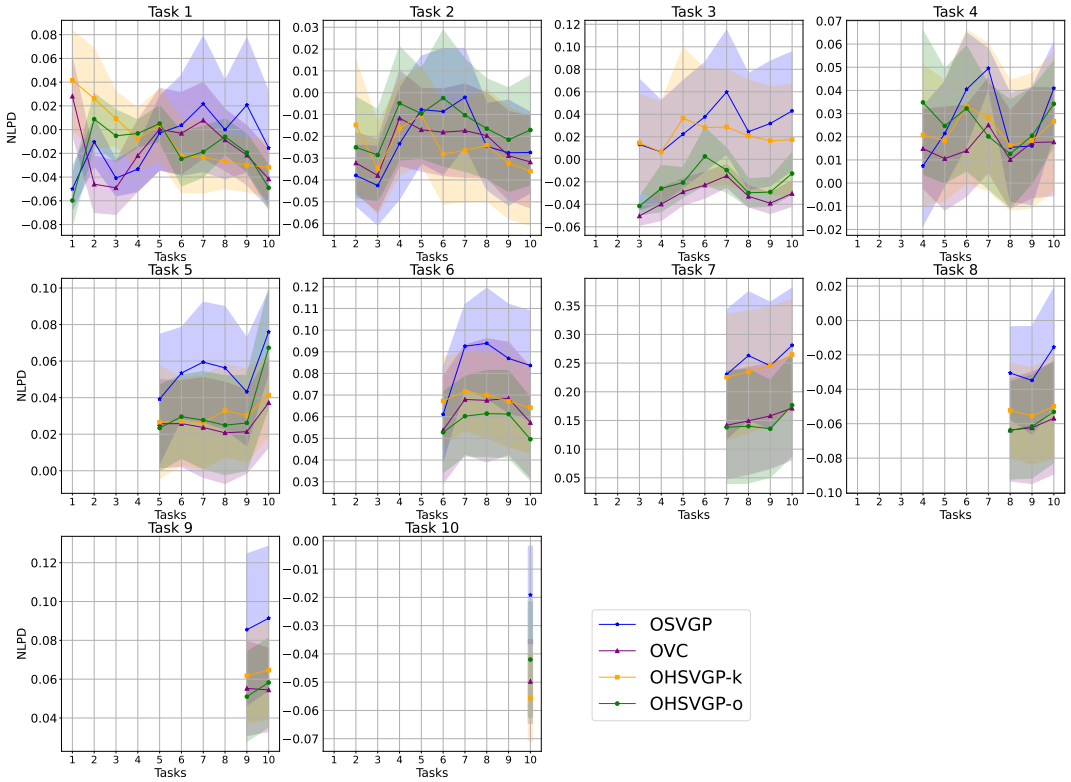


(b) Skillcraft (L2)

Figure 4.9: Test set RMSE per task after continually learning each task for all the 10 tasks on UCI Skillcraft dataset.

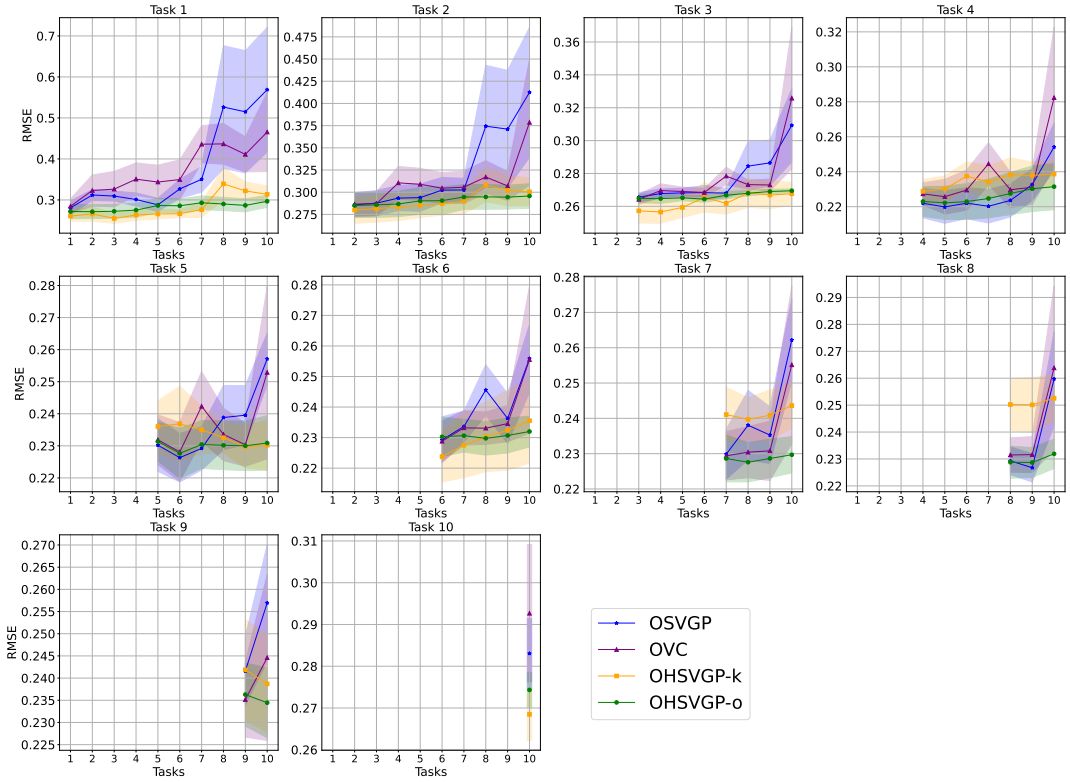


(a) Powerplant (1st dimension)

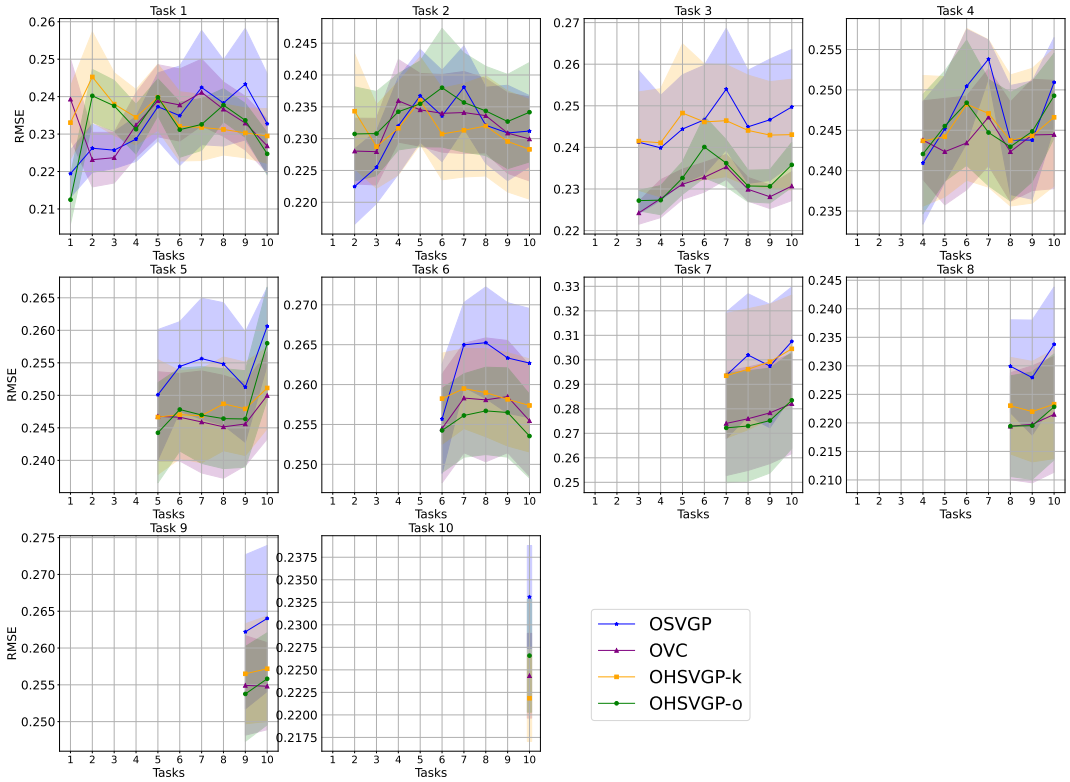


(b) Powerplant (L2)

Figure 4.10: Test set NLPD per task after continually learning each task for all the 10 tasks on UCI Powerplant dataset.



(a) Powerplant (1st dimension)

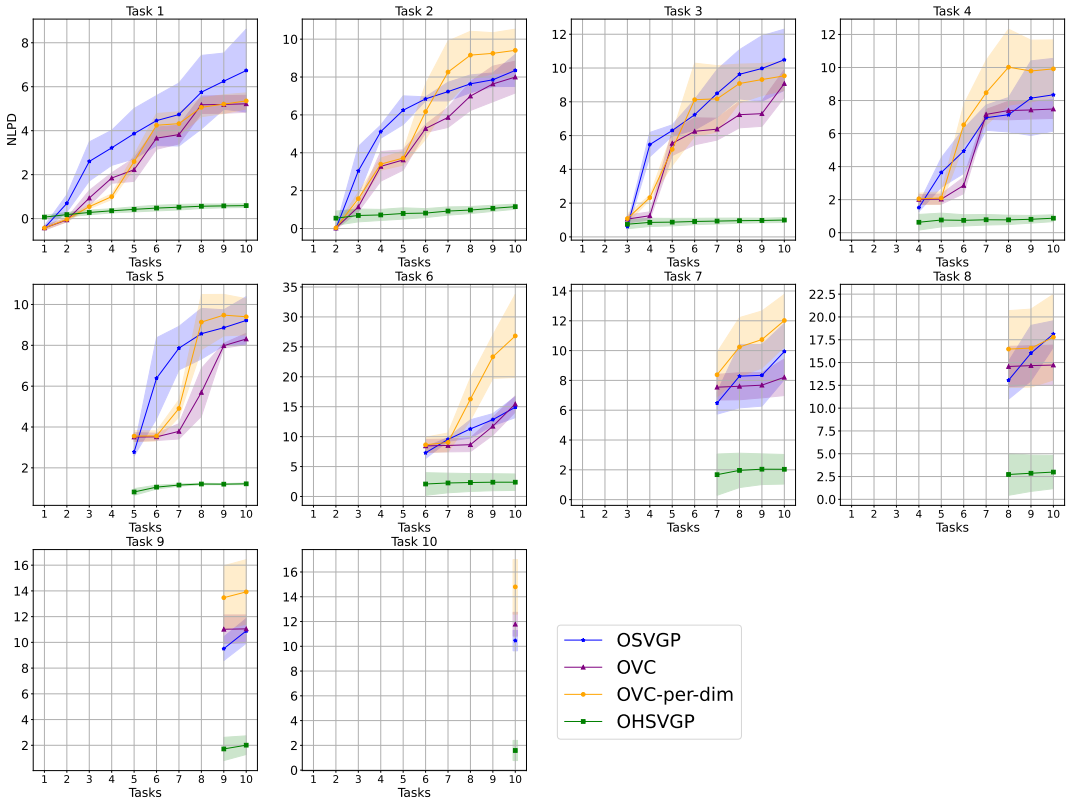


(b) Powerplant (L2)

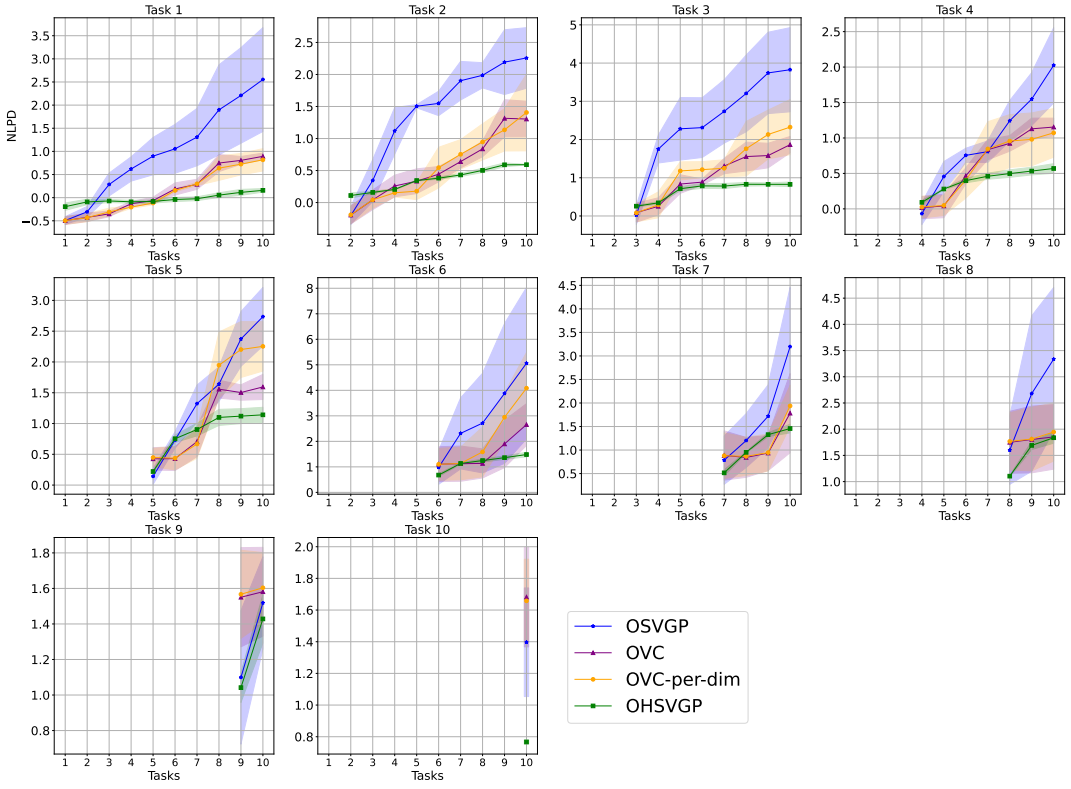
Figure 4.11: Test set RMSE per task after continually learning each task for all the 10 tasks on UCI Powerplant dataset.

4.6.3 Continual Learning for High Dimensional Time Series Prediction

All models share a two-layer MLP encoder–decoder, a 20-dimensional latent space, and a multi-output GP with independent components; we use $M \in \{50, 100\}$ and train each task for 20 epochs with learning rate 0.005 on a single NVIDIA A6000 GPU. The continual learning in SVGPVAE is achieved by further imposing Elastic Weight Consolidation (EWC; (Kirkpatrick et al., 2017)) loss on the encoder and decoder, which yields the vanilla baseline, Online SVGPVAE (OSVGP). Since EWC alone leaves inducing locations non-regularized, a principled online placement rule for the inducing locations will improve the model. Thus, we further consider OVC-SVGPVAE (OVC) which adjusts inducing locations online via Pivoted-Cholesky, and OVC-SVGPVAE per dimension (OVC-per-dim), which makes OVC more flexible by allocating a separate set of M inducing points to every latent dimension. Our method, Online HiPPO SVGPVAE (OHSVGP), replaces standard inducing points in SVGPVAE with HiPPO inducing variables and updates them online via recurrence. Figure 4.12 and Figure 4.13 plot the change of NLPD and RMSE during continual learning for all tasks, respectively. The performance of OHSVGP remains stable throughout, while the other methods all demonstrate obvious catastrophic forgetting shown by the large gaps between performances after learning current task i and after learning final task 10. Two factors plausibly explain the gap: first, standard inducing points cannot adequately cover the long time axis, whereas OHSVGP ties its inducing variables to basis functions rather than time locations; second, the added encoder–decoder complexity makes optimization harder for models that must reuse a limited inducing set. Increasing M narrows the gap but scales at $\mathcal{O}(M^3)$ computational and $\mathcal{O}(M^2)$ memory cost respectively, underscoring OHSVGP's superior efficiency.

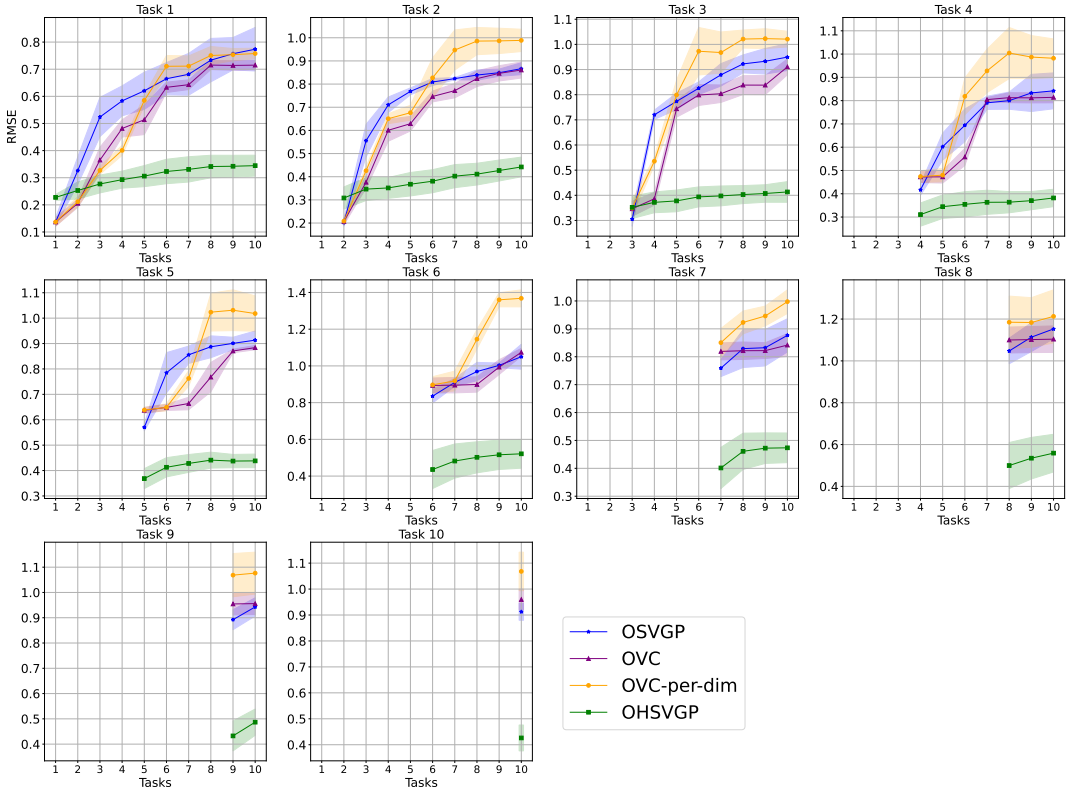


(a) M=50

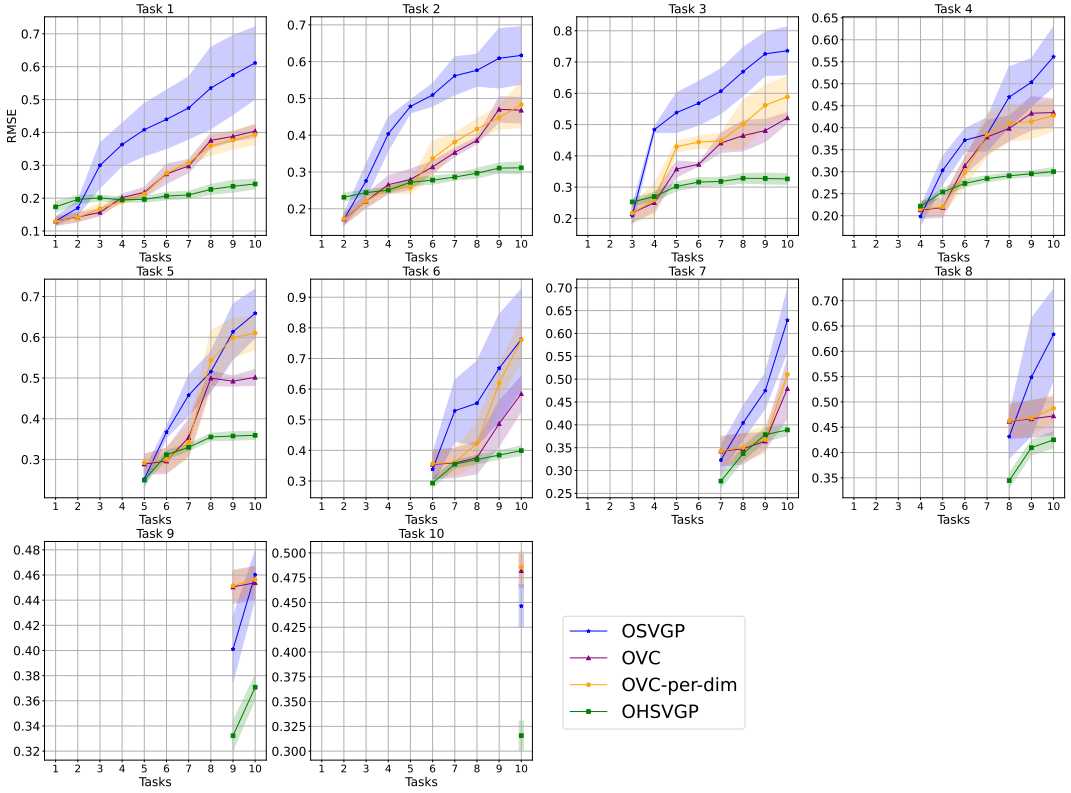


(b) M=100

Figure 4.12: Test set NLPD per task after continually learning each task for all the 10 tasks on ERA5 dataset.



(a) M=50



(b) M=100

Figure 4.13: Test set RMSE per task after continually learning each task for all the 10 tasks on ERA5 dataset.

4.7 Conclusion

We introduce OHSVGP, a novel online Gaussian process model that leverages the HiPPO framework for robust long-range memory in online/continual learning. By interpreting HiPPO's time-varying orthogonal projections as adaptive interdomain GP basis functions, we leverage SSM for improved online GP. This connection allows OHSVGP to harness HiPPO's efficient ODE-based recurrent updates while preserving GP-based uncertainty-aware prediction. Empirical results on a suite of online and continual learning tasks show that OHSVGP outperforms existing online sparse GP methods, especially in scenarios requiring long-term memory. Moreover, its recurrence-based covariance updates yield far lower computational overhead than OSVGP's sequential inducing point optimization. This efficient streaming capability and preservation of historical information make OHSVGP well-suited for real-world applications demanding both speed and accuracy.

Chapter 5

Your Image is Secretly the Last Frame of a Pseudo Video

This chapter is based on Chen et al. (2025a):

- Wenlong Chen*, Wenlin Chen*, Lapo Rastrelli, and Yingzhen Li. **Your Image is Secretly the Last Frame of a Pseudo Video**. In *Deep Generative Model in Machine Learning: Theory, Principle and Efficacy (DeLTa) Workshop at ICLR*, 2025.

The main idea was developed by me while the last author provided useful suggestions. Both co-first authors (∗) contributed to code implementation, experimentation, and paper writing under the supervision of the last author. The third author helped with the experiments related to video diffusion models. I also helped orchestrate other authors' contributions.

In the previous two chapters, we combined techniques from deep sequence modeling and probabilistic methods to build powerful predictive models. This chapter explores sequence modeling in the orthogonal domain of deep generative models. Our key observation is that one of the main differences between two classes of sequential generative models, diffusion models and hierarchical variational encoders, is that diffusion models incorporate inductive bias as direct supervision signals for their sequence of intermediate latent variables. Inspired by this observation and the empirical success of diffusion models, we propose to repeatedly apply data augmentation to images to create pseudo-video sequences and explore the possibilities of improving other image-generative models by jointly modeling the distribution of the sequence consisting of the original image and its corresponding pseudo video containing self-supervised information.

5.1 Introduction

Sequential models form a popular framework for generating images (Gulrajani et al., 2017; Sønderby et al., 2016; Ho et al., 2020; Liu et al., 2022; Albergo et al., 2023; Lipman et al., 2023; Shi et al., 2023; Wang et al., 2024). Instead of generating images from noise in one shot, which can be challenging, these models gradually transform noise into images using multiple intermediate steps. Among them, diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021b) and their variants (Kingma et al., 2021; Nichol and Dhariwal, 2021; Song et al., 2021a; Rissanen et al., 2023; Bansal et al., 2023; Hoogeboom and Salimans, 2023) have shown impressive ability to generate high quality images in recent years.

While diffusion models can be viewed as a special case of a traditional sequential generative model, i.e., hierarchical variational autoencoders (HVAEs) (Sønderby et al., 2016; Maaløe et al., 2019; Vahdat and Kautz, 2020), they tend to outperform standard HVAEs significantly in practice. The major differences between diffusion models and standard HVAEs are two-fold. First, diffusion models tend to have much more intermediate states, which may help improve the generation quality (Huang et al., 2021). Second, diffusion models incorporate exact self-supervised information for their intermediate states: they are supposed to match the corrupted (e.g., noisy or blurred) versions of the original target image at different corruption levels. These additional information helps regularize training and guide generation in diffusion models. On the other hand, the intermediate states in standard HVAEs are unobserved and one does not have explicit control of them. Consequently, there may be many different distributions over the intermediate states that are capable of generating images (i.e., the issue of unidentifiability (Locatello et al., 2019; Khemakhem et al., 2020)). The lack of identifiability of the intermediate states may pose challenges to the optimization during training since it suggests a huge hypothesis space with many sub-optimal solutions.

In this paper, we hypothesize that incorporating such self-supervised information into flexible generative models, as in diffusion models, may be one of the key reasons that they achieve good generation performance. Based on this assumption, we explore the possibility of improving other types of image generative models by extending them to video generative models and artificially injecting self-supervised information in the form of pseudo videos whose frames are created by applying data augmentation to the original images. These pseudo videos are then used to train our video generative models. After that, we compare the generation quality of the last frame of the pseudo video (corresponding to the original image) generated by the video generative model with that of the images generated by the original image

generative model. Empirically, we observe improved image generation quality via pseudo video generation compared to the images directly generated by the original image generative model. Theoretically, we provide intuitions on why designing better pseudo videos with data augmentation beyond first-order Markov chains can be helpful.

The contributions of our paper are summarized below.

- (Section 5.2) Our key insight is that pseudo videos created by corrupting the original target image may provide useful self-supervised information for training generative models. This is demonstrated by a comparison between diffusion models and standard HVAEs as a motivating example.

- (Sections 5.3 and 5.4) We attempt to improve two popular generative model frameworks, VQVAE (Van Den Oord et al., 2017) and Improved DDPM (Nichol and Dhariwal, 2021), by extending them to their video generative model counterparts and training them on pseudo videos. Empirically, we show that this procedure improves the image generation quality with pseudo videos of just a few frames. In general, our proposed framework provides a new way of scaling up image generative models with their video generative model counterparts for potential performance gain.

- (Section 5.4) Theoretically, we analyze the potential issue of certain pseudo videos, including those in the form of first-order Markov chains, in autoregressive video generation frameworks. Based on our theoretical results, we propose a simple and effective approach which avoids the potential issues by constructing higher-order Markov pseudo videos.

5.2 Motivation

Let $\boldsymbol{x} \in \mathbb{R}^n$ (also denoted as $\boldsymbol{z}_0$) be an observed variable of interest. The task of generative modeling aims to fit a parametric model $p_\theta(\boldsymbol{x})$ to estimate the data distribution $p(\boldsymbol{x})$ using samples from $p(\boldsymbol{x})$. We have already reviewed hierarchical variational autoencoders (HVAEs; (Sønderby et al., 2016; Maaløe et al., 2019; Vahdat and Kautz, 2020)) and diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021b) in Section 2.4.2 and Section 2.4.3, respectively, and we refer the readers to these two sections if a refresh of the related background and notation is needed. Our key observation is obtained from the difference between the training objectives for these two classes of deep sequential generative models:

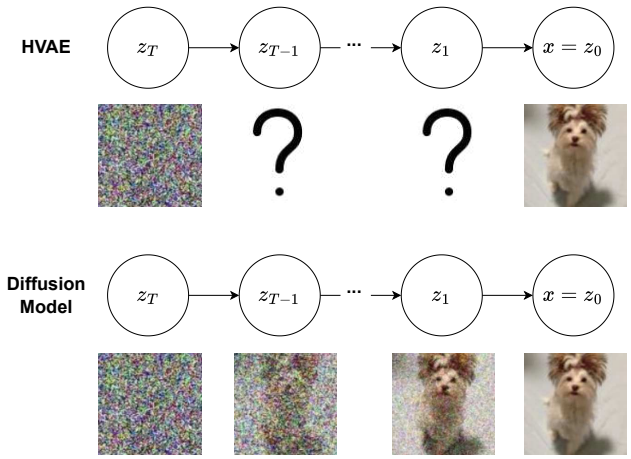


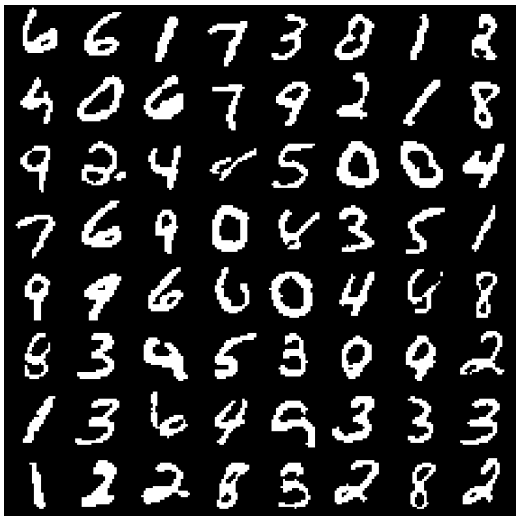
Figure 5.1: Diffusion model vs HVAE: compared to standard HVAEs, diffusion models incorporate inductive bias for its intermediate latent states with self-supervised signals.

$$\mathcal{L}_{ELBO}^{\text{HVAE}}(\boldsymbol{x}, \boldsymbol{\theta}, q_{\boldsymbol{\phi}}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{z}_0)}\left[\log \frac{p(\boldsymbol{z}_T)\prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)}{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{z}_0)}\right], \tag{5.1}$$

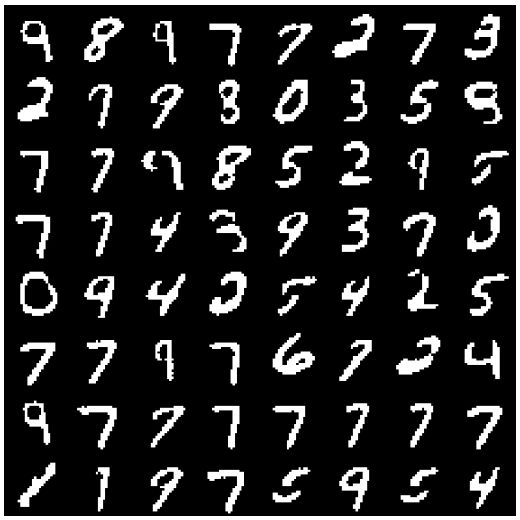
$$\begin{aligned}\mathcal{L}_{ELBO}^{\text{Diffusion}}(\boldsymbol{x}, \boldsymbol{\theta}) &= \mathbb{E}_{q(\boldsymbol{z}_{1:T}|\boldsymbol{z}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{z}_0|\boldsymbol{z}_1) - \sum_{t=1}^{T} \text{KL}(q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{z}_0)||p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}))\right] \\ &= \mathbb{E}_{q(\boldsymbol{z}_{1:T}|\boldsymbol{z}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{z}_0|\boldsymbol{z}_1) - \sum_{t=1}^{T} \underbrace{\frac{\|\mu_{\boldsymbol{\theta}}(\boldsymbol{z}_t, t) - \tilde{\mu}(\boldsymbol{z}_t, \boldsymbol{z}_0)\|^2}{2\sigma_t^2}}_{\text{mean matching}}\right].\end{aligned} \tag{5.2}$$

5.2.1 Diffusion Model vs Hierarchical VAE

Compared to the objective for training standard HVAEs (Eq. 5.1), one can see that the objective for training diffusion models (Eq. 5.2) incorporates direct control for the intermediate states. Due to the fixed pre-defined inference model (Eq. 2.58), the objective in Eq. 5.2 is simplified. In particular, its second term (mean matching) suggests that at each intermediate step t, the mean function $\mu_\theta(\boldsymbol{z}_t, t)$ in the generation model is trained by matching a noisy version $\tilde{\mu}(\boldsymbol{z}_t, \boldsymbol{z}_0)$ of the original target image $\boldsymbol{z}_0 := \boldsymbol{x}$. In contrast, the objective for standard HVAEs has no such information to impose any control over their intermediate states since their inference models are parameterized by flexible neural networks and keep being updated along with the generation model by end-to-end training. Figure 5.1 illustrates this difference



(a) HVAE with heat equation encoder



(b) Standard HVAE with learnable encoder

Figure 5.2: Generated digits from HVAE with encoder fixed according to the heat equation and standard HVAE with learnable encoder. Both HVAEs use the same decoder architecture as in Rissanen et al. (2023)

between diffusion models and standard HVAEs. Without the aid of the self-supervised information, the intermediate states in standard HVAEs are very flexible, which implies that they are unidentifiable in the sense that there could be many plausible distributions over them that can generate images (i.e., many sub-optimal solutions), which makes the optimization harder as T becomes larger. In contrast, diffusion models may benefit from the self-supervised information (i.e., noisy images) for their intermediate states, for which optimization can be less challenging even with large T, since this inductive bias pins down one specific route of generation, which eliminates other solutions that are inconsistent with this inductive bias.

Table 5.1: Inception Score (IS) of generated digits from HVAE with encoder fixed according to the heat equation and standard HVAE with learnable encoder.

	HVAE with heat equation encoder	Standard HVAE with learnable encoder
IS	**9.32**	7.27

To show the critical role of self-supervised information, we train an HVAE with a similar architecture as in Rissanen et al. (2023) on the binarized MNIST dataset (Salimans et al., 2015) as a proof of concept, where the encoder is fixed according to the heat equation (Rissanen et al., 2023):

$$q(\boldsymbol{z}_{1:T}|\boldsymbol{z}_0) = \prod_{t=1}^{T} q(\boldsymbol{z}_t|\boldsymbol{z}_0) = \prod_{t=1}^{T} \mathcal{N}(\boldsymbol{z}_t|F_h(t)\boldsymbol{z}_0, \sigma_h^2), \tag{5.3}$$

where F_h is the matrix for simulating the heat equation until time t. This can

be seen as an HVAE trained with explicit supervision signals from pseudo videos created by the heat equation. We create $T = 18$ frames of pseudo videos for each training image. Figure 5.2 demonstrates that the HVAE trained with pseudo videos created by the heat equation can generate much sharper and diverse digits than a standard HVAE which uses the same architecture but with a learnable encoder. We also report in Table 5.1 the Inception Score (IS) of the generated images for quantitative comparison and HVAE trained with pseudo videos created by the heat equation achieves noticeably higher IS than standard HVAE. Here, we deliberately use heat equation instead of Gaussian noise to create pseudo frames to show that there are plenty of choices to create pseudo videos that contain useful self-supervised information besides adding Gaussian noise as in standard diffusion models.

5.2.2 Improving Image Generation via Pseudo Video Generation

With the concept of pseudo videos and their effectiveness in diffusion models, we are interested in the following open generic question in this work:

> *Is it possible to improve other types of image generative models by jointly modelling the distribution of the original image and its corresponding pseudo video which contains self-supervised information?*

The answer is affirmative. In this work, we show empirical evidence of the advantages of pseudo videos on two types of generation models, namely improving VQVAE (Van Den Oord et al., 2017) (Section 5.3) and DDPM (Nichol and Dhariwal, 2021) (Section 5.4.2) with Phenaki (Villegas et al., 2022) and Video Diffusion (Harvey et al., 2022) trained on pseudo videos, respectively. Moreover, we provide theoretical arguments favouring the use of more expressive ways of creating pseudo videos in the autoregressive video generation framework (Section 5.4.1), beyond the first-order Markov strategy as typically used in the forward process of standard diffusion models.

5.3 Improved Reconstruction and Generation in VQ-VAE with Pseudo Videos

In this section, we utilize pseudo videos to improve image generation quality of Vector Quantized Variational Autoencoder (VQVAE) (Van Den Oord et al., 2017), where the latent variables $\boldsymbol{z}$ are discrete tokens.

5.3.1 Preliminaries of VQVAE

Vector Quantized Variational Autoencoder (VQVAE), similar to VAE, is also a deep latent variable model assuming a data generative process of the form:

$$\boldsymbol{x} \sim \int p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{z}) d\boldsymbol{z}. \tag{5.4}$$

The differences of VQVAE from VAE are two folds. First, while VAE assumes $\boldsymbol{z}$ to be continuous and $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ to be a standard Gaussian apriori, VQVAE employs discrete latent variables $z \in \{1, \cdots, K\}$, and parameterizes $p_{\boldsymbol{\theta}}(z)$ with another deep neural network. Second, while VAE trains the generative model parameters $\boldsymbol{\theta}$ associated with the likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ via approximate MLE, VQVAE follows a two-step training procedure by fitting the likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x}|z)$, and prior $p_{\boldsymbol{\theta}}(z)$, separately. Given a training set of observations $\{\boldsymbol{x}_n\}_{n=1}^N$, the first step is learning the conditional distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}|z)$, and it is achieved with an auxiliary encoder, $z_{\boldsymbol{\phi}}(\boldsymbol{x}) : \mathcal{X} \to \{1, \cdots, K\}$, (similar to the concept of inference model in VAE):

$$z_{\boldsymbol{\phi}}(\boldsymbol{x}) = \underset{i \in \{1, \cdots, K\}}{\arg\min} ||f_{\boldsymbol{\phi}}(\boldsymbol{x}) - \boldsymbol{z}_i^{\text{emb}}||, \tag{5.5}$$

where $f_\phi(\cdot) : \mathcal{X} \to \mathbb{R}^d$ is parameterized by a neural network and $\boldsymbol{z}_i^{\text{emb}} \in \mathbb{R}^d$ is a d-dimensional latent embedding associated with discrete code i. We then train the encoder and the likelihood jointly with the following loss for each observation:

$$\begin{aligned} \mathcal{L}_{VQVAE}(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\phi}, \{\boldsymbol{z}_i\}_{i=1}^K) = &- \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|k) + ||sg(f_{\boldsymbol{\phi}}(\boldsymbol{x})) - \boldsymbol{z}_k^{\text{emb}}||_2^2 \\ &+ \beta ||f_{\boldsymbol{\phi}}(\boldsymbol{x}) - sg(\boldsymbol{z}_k^{\text{emb}})||_2^2, \quad k = z_{\boldsymbol{\phi}}(\boldsymbol{x}) \end{aligned} \tag{5.6}$$

where $sg(\cdot)$ denotes the "stop gradient" operator. The first term in the objective is the negative log-likelihood which encourages accurate reconstruction, and the second term is the quantization loss, ensuring that the latent embedding is close to the encoder network outputs $f_\phi(\boldsymbol{x})$, and the third term is the commitment loss which makes sure the update for encoder parameters is not too fast when $\beta < 1$, so that the encoder can commit to an embedding.

Note that the latent space can be represented by multi-dimensional discrete tokens. For example, for an RGB image $\boldsymbol{x}$ with size $H \times W \times 3$, $f_\phi(\cdot)$ is typically parameterized by a convolutional neural network which maps $\boldsymbol{x}$ to an embedding with size $h \times w \times d$, and each dimension of $f_{\boldsymbol{\phi}}(\boldsymbol{x})$ is quantized into one of K discrete tokens as in Eq. 5.5, so that $\boldsymbol{z}_{\boldsymbol{\phi}}(\boldsymbol{x})$ is a tensor of size $h \times w$ with discrete elements.

In the second step of learning $p_{\boldsymbol{\theta}}(\boldsymbol{z})$, we map each $\boldsymbol{x}$ in the training set into latent discrete tokens, which forms an empirical distribution of the discrete latent variables,

$p(\boldsymbol{z}) \approx p_{\text{empirical}}(\boldsymbol{z}) = \frac{1}{N}\sum_{n=1}^{N}\delta_{\boldsymbol{z}_{\phi}(\boldsymbol{x}_n)}(\boldsymbol{z})$. Next, we can choose a particular factorization order for $p_{\text{empirical}}(\boldsymbol{z})$, and fit an autoregressive model (van den Oord et al., 2016a,b), $p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_1, \cdots, \boldsymbol{z}_{t-1})$, to predict the next discrete code given the sequence of previously generated tokens by maximum likelihood, with training set obtained from $p_{\text{empirical}}(\boldsymbol{z})$. The final prior model is then $p_{\boldsymbol{\theta}}(\boldsymbol{z}) = p_{\boldsymbol{\theta}}(\boldsymbol{z}_1)p_{\boldsymbol{\theta}}(\boldsymbol{z}_2|\boldsymbol{z}_1)\cdots(\boldsymbol{z}_T|\boldsymbol{z}_{t<T_z})$, where T_z is the total number of discrete tokens in $\boldsymbol{z}$ (e.g., $T_z = h \times w$ in the above image-generative example).

5.3.2 Experiments

To incorporate the pseudo video sequences into the model, we employ a video generative model counterpart of VQVAE, C-ViViT (Villegas et al., 2022), to compress the pseudo videos of size $T \times H \times W \times 3$ into latent discrete tokens of size $t \times h \times w$. Instead of convolutional neural network, C-ViViT includes temporal transformer layers to handle the temporal dimension and in its hidden layers, the height and width dimension are flatten into a single "spatial dimension" of size $h \times w$, and C-ViViT also considers spatial transformer layers to compute attention along the spatial dimension. In addition to standard VQVAE loss defined in Eq. 5.6, C-ViViT also adds a GAN style adversarial loss (Karras et al., 2020) and an image perceptual loss (Johnson et al., 2016; Zhang et al., 2018b) to the final objective. For the latent space created by a C-ViViT, we consider two generative models to fit a prior $p(\boldsymbol{z})$ for the latent tokens:

- VideoGPT (Yan et al., 2021) uses an autoregressive (AR) Transformer (Brown et al., 2020) to factorize $p(\boldsymbol{z}) = \prod_{i=1}^{T_z} p(\boldsymbol{z}_i|\boldsymbol{z}_{<i})$ in an autoregressive manner with masked self-attention, where T_z is the total number of the tokens, and is trained with maximum likelihood. VideoGPT is the video generative model counterpart extended from ImageGPT (Chen et al., 2020a).
- Phenaki (Villegas et al., 2022) uses a bidirectional Transformer (Vaswani et al., 2017) to predict all tokens in one shot rather than in an autoregressive manner. At each training step, one samples a masking ratio $\gamma \in (0, 1)$, and the model is trained by predicting the masked tokens given the unmasked ones. During generation, all tokens are masked initially, and the model predicts all tokens simultaneously. The generation will then be refined following a few steps of re-masking and re-prediction, with a decreasing masking ratio as we proceed. Phenaki is the video generative model counterpart extended from MaskGit (Chang et al., 2022).

We compare the generation quality of the last frames (corresponding to the original images) in the generated videos from the video generative model trained on pseudo

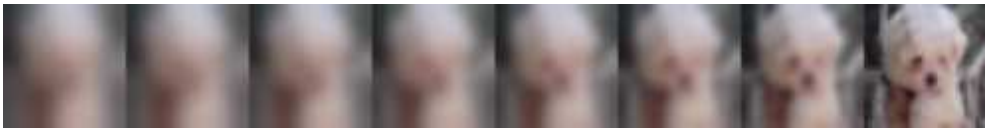


Figure 5.3: An example of pseudo video constructed by transforming an image of a dog using blurring.

videos to the images generated by the original image generative model trained on the original target images. We include experimental setups below and we also include detailed hyperparameters in Appendix D.2.

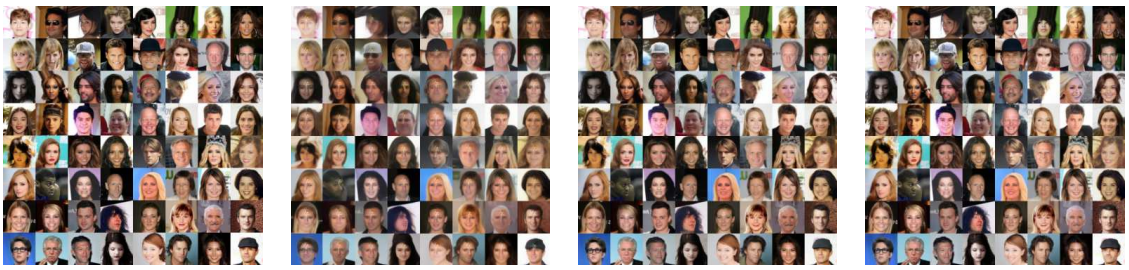
Datasets. We create 8-frame and 18-frame pseudo videos using images from two benchmark datasets, CIFAR10 (32×32) (Krizhevsky et al., 2009) and CelebA (64×64) (Liu et al., 2015), with the blurring technique from Bansal et al. (2023). To create 8-frame pseudo videos, we blur the images recursively 7 times with a Gaussian kernel of size 11×11 and standard deviation growing exponentially at the rate of 0.05. For 18-frame pseudo videos, we blur the images 17 times with same Gaussian kernel but with a standard deviation growing exponentially at the rate of 0.01. The pseudo videos are organized such that the last frames are the original target images and the first frames are the blurriest images. Figure 5.3 shows an example of such a pseudo video. We use 1-frame to denote images generated by the original image VQVAE model trained on the original target images. Initially, we also tried using Gaussian noise as data augmentation but we did not manage to obtain decent results. We believe it might be because VQVAE and C-ViViT compress the data to discrete latent tokens, and if using Gaussian noise as data augmentation, it will increase the variability of the pixel values in each patch. Therefore it would be hard to compress the pseudo videos with so much variability into discrete tokens defined with finite codebook size. In contrast, blurring will reduce the variability in the image and make the pixels close to each other have similar values, which makes it easier for a VQVAE or C-ViViT with finite codebook size to compress the pseudo videos.

Network Architectures.[1] We train VQVAE based generative models with a codebook size (K) of 1024 for the discrete latent tokens. Specifically, we use C-ViViT as compression model (autoencoder) for pseudo videos. For a pseudo video with shape $(T, H, W, 3)$, we compress it to discrete latent tokens with shape $(\frac{T}{2}, \frac{H}{4}, \frac{W}{4})$ by extracting video patches of size $2 \times 4 \times 4$. For images with shape $(1, H, W, 3)$, we use VQVAE to compress them to tokens with shape $(1, \frac{H}{4}, \frac{W}{4})$ by extracting image patches of size 4×4. We then consider two video generative models, VideoGPT and

[1]For 1-frame models, we have tried using deeper architectures but observed no improvement in performance, which suggests that our pseudo video framework can help further improve the performance of generative models while simply increasing the model size becomes ineffective.

(a) Ground-truth (b) 1-frame reconstruction (c) 8-frame reconstruction (d) 18-frame reconstruction

Figure 5.4: Ground-truth images and reconstructed images from VQVAE/CViViT trained on CIFAR10.



(a) Ground-truth (b) 1-frame reconstruction (c) 8-frame reconstruction (d) 18-frame reconstruction

Figure 5.5: Ground-truth images and reconstructed images from VQVAE/CViViT trained on CelebA.

Phenaki (and their image generative model counterparts, ImageGPT and MaskGit), to fit the prior over the latent tokens.

- **VQVAE/C-ViViT (reconstruction).** We use a similar architecture as in Villegas et al. (2022), which has a 4-layer spatial Transformer, a 4-layer temporal Transformer, with a hidden dimension of 512. For 1-frame VQVAE model, we consider a 8-layer spatial Transformer since there is no temporal dimension.

- **ImageGPT/VideoGPT (AR generation).** We use a similar architecture as in Yan et al. (2021), which has a 8-layer autoregressive (AR) Transformer with 4 attention heads and a hidden dimension of 144.

- **MaskGit/Phenaki (latent masked generation).** We use a similar architecture as in Villegas et al. (2022), which has a 6-layer bidirectional Transformer with a hidden dimension of 512.

Evaluation Metric. We compute Frechet Inception Distance (FID ↓) (Heusel et al., 2017) with 50k samples to evaluate the quality of images, either from the last-frames of videos generated by video models trained on pseudo videos, or images

Table 5.2: Last-frame FID/PSNR of images produced by C-ViViT (reconstruction), VideoGPT (AR generation) and Phenaki (latent masked generation) trained on pseudo videos constructed from CIFAR10 and CelebA images. 1-frame results are obtained from their image counterparts VQVAE (reconstruction), ImageGPT (AR generation) and MaskGit (latent masked generation) trained on original CIFAR10 and CelebA images.

	CIFAR10			CelebA		
	1-frame	8-frame	18-frame	1-frame	8-frame	18-frame
Reconstruction (FID)	24.53	13.81	**11.26**	24.62	5.72	**2.27**
Reconstruction (PSNR)	19.97	21.86	**25.89**	20.52	24.66	**29.68**
AR Generation (FID)	72.06	**54.60**	69.23	32.98	30.19	**28.08**
Latent Masked Generation (FID)	49.27	**35.50**	47.65	27.34	16.87	**16.66**

generated by image models trained on the original target images. We also report Peak Signal-to-noise Ratio (PSNR ↑) as an additional metric for reconstruction result.

Results. Table 5.2 shows the last-frame FID and PSNR of C-ViViT for reconstruction and that of VideoGPT and Phenaki for AR and masked generation on CIFAR10 and CelebA, respectively. The 1-frame results correspond to the performance of their image model counterparts (i.e., VQVAE for image reconstruction, and ImageGPT and MaskGit for AR and masked image generation, respectively). We observe that pseudo videos indeed help improve the training of the C-ViViT as the reconstruction quality of the last frame is significantly improved with a few more frames. We show reconstructed CIFAR10 and CelebA images from different C-ViViT models in Figures 5.4 and 5.5, respectively. It can be seen that for both datasets, the reconstructed images trained with 1-frame models are relatively smoothed and this issue is resolved by using pseudo videos which produce sharper images. Quantitatively, reconstruction FID improves as more frames are used. Pseudo videos also improve the image generation performance compared to the 1-frame results. Interestingly, we see a diminishing return as we include more frames. For CelebA images, 18-frame pseudo videos help achieve the best image generation performance for both AR and latent masked generation, but 8-frame models achieve a comparable performance. For CIFAR10 images, 8-frame pseudo videos result in better image generation performance than 18-frame pseudo videos, which suggests the latent tokens have a more complex prior distribution in order to reconstruct pseudo videos with 18 frames well and therefore this prior is more difficult for VideoGPT or Phenaki to capture. In summary. while pseudo videos help improve the performance, the optimal number of frames may depend on the dataset and the augmentation strategy. This diminishing return is not a severe issue in practice since practitioners may prefer to improve

the generation with just a few more pseudo frames to avoid introducing a high computational cost.

5.4 Improved Generation via Higher-order Markov Pseudo Videos

Since pseudo video contains extra information on the target image, we would like to better understand what type of additional information can be leveraged to achieve better image generation quality. In practice, since there are infinitely many data augmentation strategies to create pseudo videos, we would like to study which types of data augmentation are more favorable to shed light on the practical design of pseudo videos.

5.4.1 Is First-order Markov Chain the Optimal Choice for Creating Pseudo Videos?

Consider pseudo video $\boldsymbol{z}_{1:T}$, where $\boldsymbol{z}_T := \boldsymbol{x}$ is the target image[2], and $\boldsymbol{z}_t$'s ($t < T$) are some noisy measurements of $\boldsymbol{z}_T$ created with some data augmentation. We show that in principle, generative models that utilize more pseudo frames to generate $\boldsymbol{z}_T$ are more likely to achieve better performance, and passing information of the target image to the pseudo frames with a first-order Markov chain as in standard diffusion models may not be the optimal choice. We demonstrate it with the following autoregressive video generation example.

Consider building a generative model g that predicts $\boldsymbol{z}_T$ by taking advantage of the information in $\boldsymbol{z}_{T-1}$ alone. We train the model by minimizing the reconstruction error. The minimum of this loss is

$$\mathcal{L}_1^* = \min_g \mathbb{E}_{p(\boldsymbol{z}_T, \boldsymbol{z}_{T-1})}[||\boldsymbol{z}_T - g(\boldsymbol{z}_{T-1})||_2^2] = \mathbb{E}_{p(\boldsymbol{z}_{T-1})}[\mathrm{Var}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{T-1})}(\boldsymbol{z}_T)], \tag{5.7}$$

which is achieved at the non-parametric optimum $g^*(\boldsymbol{z}_{T-1}) = \mathbb{E}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{T-1})}[\boldsymbol{z}_T]$, where $\mathrm{Var}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{T-1})}(\boldsymbol{z}_T) = \mathbb{E}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{T-1})}\{(\boldsymbol{z}_T - \mathbb{E}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{T-1})}[\boldsymbol{z}_T])^\top(\boldsymbol{z}_T - \mathbb{E}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{T-1})}[\boldsymbol{z}_T])\}$. Now consider another model h that predicts $\boldsymbol{z}_T$ using both $\boldsymbol{z}_{T-1}$ and $\boldsymbol{z}_{T-2}$ by minimizing the reconstruction error again. The minimum reconstruction error this time is

$$\mathcal{L}_2^* = \min_h \mathbb{E}_{p(\boldsymbol{z}_T, \boldsymbol{z}_{T-1}, \boldsymbol{z}_{T-2})}[||\boldsymbol{z}_T - h(\boldsymbol{z}_{T-1}, \boldsymbol{z}_{T-2})||_2^2] = \mathbb{E}_{p(\boldsymbol{z}_{T-1}, \boldsymbol{z}_{T-2})}[\mathrm{Var}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{T-1}, \boldsymbol{z}_{T-2})}(\boldsymbol{z}_T)], \tag{5.8}$$

[2]Unlike diffusion models where $\boldsymbol{z}_0$ denotes the original image, from here onwards we will denote the original image by $\boldsymbol{z}_T$ since it is the last frame of the pseudo video.

which is achieved at the non-parametric optimum $h^*(\boldsymbol{z}_{T-1}, \boldsymbol{z}_{T-2}) = \mathbb{E}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{T-1},\boldsymbol{z}_{T-2})}[\boldsymbol{z}_T]$. The benefit of using more pseudo frames to generate the target image can be seen from the fact that the minimum reconstruction error will never increase by using more pseudo frames since by the law of total variance,

$$\mathcal{L}_2^* - \mathcal{L}_1^* = -\mathbb{E}_{p(\boldsymbol{z}_{T-1})}\{\mathrm{Var}_{p(\boldsymbol{z}_{T-2}|\boldsymbol{z}_{T-1})}(\mathbb{E}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{T-1},\boldsymbol{z}_{T-2})}[\boldsymbol{z}_T])\} \leq 0. \tag{5.9}$$

Moreover, the non-optimality of creating pseudo video via first-order Markov chain becomes clear: the first-order Markov data augmentation implies that $p(\boldsymbol{z}_T|\boldsymbol{z}_{T-1}) = p(\boldsymbol{z}_T|\boldsymbol{z}_{T-1}, \boldsymbol{z}_{T-2})$ and consequently $\mathcal{L}_2^* = \mathcal{L}_1^*$. More specifically, for strict inequality in Eq 5.9, we need to avoid $p(\boldsymbol{z}_T|\boldsymbol{z}_{T-1}) = p(\boldsymbol{z}_T|\boldsymbol{z}_{T-1}, \boldsymbol{z}_{T-2})$, which is equivalent to avoiding the use of either first-order Markov chain $\boldsymbol{z}_T \to \boldsymbol{z}_{T-1} \to \boldsymbol{z}_{T-2}$ or $\boldsymbol{z}_T \leftarrow \boldsymbol{z}_{T-1} \to \boldsymbol{z}_{T-2}$. This analysis is informative for us to design better pseudo videos, for example through data augmentation with higher-order Markov chains. We formalize the above informal reasoning into Theorem 5.4.1 and provide the formal proof in Appendix D.1.

Theorem 5.4.1. *Consider two video generative models that predict the last-frame $\boldsymbol{z}_T$ based on some previous frames. Suppose that they take the form of $\hat{\boldsymbol{z}}_T^{(g)} = g(\boldsymbol{z}_{s_1}, \boldsymbol{z}_{s_2}, \cdots, \boldsymbol{z}_{s_k})$ and $\hat{\boldsymbol{z}}_T^{(h)} = h(\boldsymbol{z}_{s_1}, \boldsymbol{z}_{s_2}, \cdots, \boldsymbol{z}_{s_l})$, respectively, where $T > s_1 > \cdots > s_k > \cdots > s_l$. Then, we have*

$$\min_{\hat{\boldsymbol{z}}_T^{(h)}} \mathbb{E}_{p(\boldsymbol{z}_T, \boldsymbol{z}_{s_1}, \cdots, \boldsymbol{z}_{s_l})}[||\boldsymbol{z}_T - \hat{\boldsymbol{z}}_T^{(h)}||_2^2] \leq \min_{\hat{\boldsymbol{z}}_T^{(g)}} \mathbb{E}_{p(\boldsymbol{z}_T, \boldsymbol{z}_{s_1}, \cdots, \boldsymbol{z}_{s_k})}[||\boldsymbol{z}_T - \hat{\boldsymbol{z}}_T^{(g)}||_2^2], \tag{5.10}$$

where the equality attains if $\boldsymbol{z}_T|\boldsymbol{z}_{s_1}, \cdots, \boldsymbol{z}_{s_k} \overset{d}{=} \boldsymbol{z}_T|\boldsymbol{z}_{s_1}, \cdots, \boldsymbol{z}_{s_l}$.

Remark 5.4.2. The minimum reconstruction errors above are obtained with non-parametric optima. In practice, this corresponds to the assumption that our neural networks g_θ and h_ϕ are flexible enough to accurately approximate the non-parametric optima for the theorem to hold. Besides, the analysis is based on the assumption that $\{\boldsymbol{z}_{s_i}\}_{i=1}^l$ are drawn from the ground-truth distribution, while in practice they also need to be generated with their associated previous frames, which means when the generated $\{\hat{\boldsymbol{z}}_{s_i}\}_{i=1}^l$ are far away from their ground-truth distribution, the theorem would not hold. Nevertheless, the analysis provides intuitions of the benefit of conditional generation with longer past contexts and the potential improvement in performance by using more expressive pseudo videos rather than the ones created with first-order Markov transition as in standard diffusion models, which we empirically verify with experiments in the following section.

5.4.2 Experiments

We consider generating each frame of the pseudo videos using the information provided in the previously generated frames. In particular, we use a video diffusion model (Harvey et al., 2022) trained by predicting frames autoregressively conditioning on the most recent previous frames in a context window of size C. In other words, the conditionals $p(\boldsymbol{z}_t|\boldsymbol{z}_{t-C:t-1})$ are estimated using a diffusion model parameterized by neural network taking $\boldsymbol{z}_{t-C:t-1}$ as an additional input. We compare the performance of video diffusion models trained on pseudo videos created by both standard first-order Markov transformation and higher-order Markov transformation to empirically verify our argument in Section 5.4.1. We describe the experimental setup below and include detailed hyperparameters in Appendix D.2.

Datasets. We create 4-frame and 8-frame pseudo videos using images from CIFAR10 (32×32) and CelebA (64×64). We use Gaussian noise as data augmentation and we consider two strategies:

- **First-order Markov.** We add Gaussian noise recursively 3 or 7 times to create first-order Markov pseudo videos with a linear schedule (Ho et al., 2020) with β ranging from 0.0001 to 0.05[3]: $\boldsymbol{z}_{T-t} = \sqrt{1-\beta_t}\boldsymbol{z}_{T-t+1} + \sqrt{\beta_t}\epsilon$, $\epsilon \sim \mathcal{N}(0, I)$.
- **High-order Markov.** While using the same noise schedule to create $\boldsymbol{z}_{T-t}$, instead of adding Gaussian noise to $\boldsymbol{z}_{T-t+1}$, we use a simple strategy to create high-order Markov pseudo videos by adding Gaussian noise to the mean of $\{\boldsymbol{z}_{T-t+s}\}_{s=1}^{t}$: $\boldsymbol{z}_{T-t} = \sqrt{1-\beta_t}[\frac{1}{t}\sum_{s=1}^{t}\boldsymbol{z}_{T-t+s}] + \sqrt{\beta_t}\epsilon$, $\epsilon \sim \mathcal{N}(0, I)$.

We plot examples of pseudo videos created with the above two strategies in Figure 5.6. We again use 1-frame to denote the results of the image generative model counterparts, improved DDPM (Nichol and Dhariwal, 2021), trained on the original target images. We also consider blurring as the data augmentation, however, its performance is worse than the performance of using Gaussian noise (see the **Results** paragraph below).

Network Architectures. We use a similar UNet architecture as in Harvey et al. (2022), with 2 residual blocks in each downsampling and upsampling layer and a base channel size of 128 across all models. Notice that Harvey et al. (2022) is built based on the same architecture as the 1-frame image diffusion model (Nichol and Dhariwal, 2021), and these hyperparameters are kept the same for the 1-frame image diffusion model. During generation, we use the "Autoreg" sampling scheme from Harvey

[3]During the hyperparameter selection stage, we experiment with maximal β selected from {0.02, 0.05, 0.1}, and 0.05 is the one that leads to the best performance with first-order Markov data augmentation.

(a) First-order

(b) High-order

Figure 5.6: Examples of a pseudo video constructed by adding Gaussian noise to a CIFAR10 image using first-order Markov chain (top) and high-order Markov chain (bottom).

Table 5.3: Last-frame FID of images generated by video diffusion models trained on pseudo videos constructed from CIFAR10 and CelebA images (with both first-order Markov or high-order Markov Gaussian noise data augmentation). 1-frame results are obtained from an image diffusion model trained on the original CIFAR10 and CelebA images with equivalent UNet architecture.

	CIFAR10			CelebA		
	1-frame	4-frame	8-frame	1-frame	4-frame	8-frame
First-order Markov	12.90	17.30	15.90	7.76	13.61	12.64
High-order Markov	12.90	**12.58**	12.80	7.76	**6.88**	7.55

et al. (2022) so that each frame $\boldsymbol{z}_t$ is generated by conditioning on the most recently generated frames in a context window, $\{\boldsymbol{z}_{t-c}\}_{c=1}^{C}$. The sizes of the context window C (i.e., the time lag) are 2 and 4 for 4-frame and 8-frame models, respectively. We consider 1,000 diffusion steps every time we generate a new frame. Since 4-frame and 8-frame models jointly generate the first 2 and the first 4 frames (the initial context window) at the beginning, respectively, they use overall 3,000 and 5,000 diffusion steps to generate the whole pseudo videos, respectively. We also consider increasing the number of diffusion steps from 1,000 to 4,000 when training the 1-frame image diffusion model and compare it with the 4-frame video diffusion models with 3,000 diffusion steps in total to ensure the performance gain in video diffusion models is not simply because we have more diffusion steps overall.

Results. We again compute FID (based on 10k samples) to evaluate the models. Table 5.3 shows the last-frame FID of pseudo videos generated by video diffusion models for CIFAR10 and CelebA images, respectively. The 1-frame results correspond to the performance of their image counterparts (i.e., improved DDPM). While video

(a) CIFAR10 (b) CelebA

Figure 5.7: Generated images from the video diffusion models trained on 4-frame high-order Markov pseudo videos of CIFAR10 and CelebA, respectively.

diffusion models trained on first-order Markov pseudo videos do not outperform the 1-frame image diffusion model, both 4-frame and 8-frame video diffusion models trained on high-order Markov pseudo videos can achieve better results on both datasets, which empirically justify the non-optimality of first-order Markov chains in terms of passing information from the target images to the pseudo frames as shown in Section 5.4.1, and our proposal of using more expressive pseudo videos rather than the ones created with first-order Markov chains. Notice that the 4-frame models outperform the 8-frame models, which may be due to the complex nature of the ground-truth distribution of longer pseudo videos, and thus more expressive architecture may be required to achieve optimal results (see Remark 5.4.2), while here we use the same UNet architecture across all models for fair practical comparison. Again, this U-turn should not be a severe issue in practice since practitioners may prefer to improve the generation with as few pseudo frames as possible to reduce additional computational cost. We visualize some generated images from the 4-frame models trained on CIFAR10 and CelebA in Figure 5.7.

Table 5.4 compares the 4-frame model with an 1-frame model but with 4,000 diffusion steps. While the performance of the 1-frame model with more overall diffusion steps improves for CIFAR10 and outperforms the 4-frame model, its performance on CelebA is worse than the 4-frame model. Moreover, on CelebA, it even becomes worse than the baseline 1-frame model with only 1,000 diffusion steps while the 4-frame video diffusion model consistently improves the performance on both datasets, which suggests simply increasing the number of diffusion steps in an image diffusion model may not always be effective.

Table 5.4: Last-frame FID of images generated by video diffusion models trained on pseudo videos constructed from CIFAR10 and CelebA images with high order Markov Gaussian noise data augmentation. 1-frame results are obtained from an image diffusion model trained on the original CIFAR10 and CelebA images. Here, the 1-frame models use 4,000 diffusion steps, while the 4-frame models use 3,000 diffusion steps overall.

	1-frame (1k steps)	1-frame (4k steps)	4-frame (3k steps overall)
CIFAR10	12.90	**11.95**	12.58
CelebA	7.76	7.87	**6.88**

Instead of using Gaussian noise, we also tried using Gaussian blur to create pseudo videos as in Section 5.3. However, our experiments on CIFAT10 with Gaussian blur suggest worse results than adding Gaussian noise (see Table 5.5), and we decided not to consider it for further experiments. This suggests that in practice the well-performed data augmentation strategies may vary across different classes of video generative models.

Table 5.5: Last-frame FID of images generated by video diffusion models trained on pseudo videos constructed from CIFAR10 images with high-order Markov data augmentation (either Gaussian noise or Gaussian blur).

	4-frame	8-frame
Gaussian noise	**12.58**	12.80
Gaussian blur	15.33	22.63

5.5 Related Work

5.5.1 Sequential Generative Models

Hierarchical variational autoencoders (HVAEs) (Sønderby et al., 2016; Maaløe et al., 2019; Vahdat and Kautz, 2020; Child, 2021; Xiao and Bamler, 2023) are a class of sequential generative models constructed by stacking standard VAEs (Kingma and Welling, 2014). Although HVAEs represent a rich class of expressive generative models, they are hard to train in practice due to optimization difficulty, as discussed in Section 5.2. Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021b; Kingma et al., 2021; Nichol and Dhariwal, 2021; Song et al., 2021a; Rissanen et al., 2023; Bansal et al., 2023; Hoogeboom and Salimans, 2023) can be seen as a special case of HVAEs where the encoders are fixed, pre-defined Gaussian convolution kernels. Specifically, they essentially regress a sequence of noisy images created from

the target image with self-supervision, as described in Section 5.2. Despite its similarity to HVAEs, diffusion models, and latent diffusion models (Rombach et al., 2022) which apply diffusion models in the lower dimensional latent space of another latent variable model (e.g., VQVAE (Van Den Oord et al., 2017)), have achieved state-of-the-art performance partially due to the additional self-supervision signal provided by the noise-corrupted images. Flow matching (Lipman et al., 2023; Liu et al., 2022; Albergo et al., 2023; Gat et al., 2024; Wang et al., 2024) is another state-of-the-art sequential generative modelling technique that trains continuous normalizing flows (Chen et al., 2018) by regressing a sequence of vector fields inducing a probability path that connects the data distribution and prior distribution with direct self-supervision. It has been show that flow matching can learn more straight trajectories than diffusion models, which requires less number of discretization steps at generation time. Furthermore, flow matching allows us to relax the Gaussian assumption for the prior distribution and thus enables coupling between arbitrary distributions (Albergo et al., 2023). In contrast, our proposed framework introduces a new family of approaches that leverage video generative models and pseudo videos with self-supervised frames to improve any given image generative models.

5.5.2 Self-supervised Learning

Self-supervised learning (Liu et al., 2021; Shwartz Ziv and LeCun, 2024) turns an unsupervised learning problem into a supervised learning problem by handcrafting pseudo labels for unlabeled data. There are two common approaches to self-supervised learning. 1) Contrastive learning (Chen et al., 2020b; Tian et al., 2020; Wu et al., 2020), predicts whether two inputs are different augmentations of the same original data. 2) Masked learning (Devlin et al., 2019; He et al., 2022; Fang et al., 2023) predicts randomly masked parts of an input given the unmasked parts. While our approach of fitting a video model to pseudo video sequences created by augmenting the original images does not belong to either of these families, it is essentially a new form of self-supervised learning since the pseudo video sequences can be seen as handcrafted pseudo labels for our model to predict, which provides the model with extra information (e.g., different fidelity of the original image).

5.6 Conclusion and Discussions

Summary. We drew our key insight from comparing standard HVAEs and diffusion models: the additional self-supervised information on the intermediate states provided by the noise corrupted pseudo frames in diffusion models may contribute to their success. Based on this insight, we proposed to leverage the self-supervised information

from the pseudo videos constructed by applying data augmentation to the target images to improve the performance of image generative models. This was done by extending image generative models to their video generative models counterparts and training video generative models on pseudo videos. We show in our experiments that for two popular image generative models, VQVAE and Improved DDPM, their video generative model counterparts trained on pseudo videos of just a few frames can improve image generation performance, which empirically verified the benefit of the additional self-supervised information in the pseudo videos.

Discussions and Future Work. Our proposed framework provides an alternative approach of scaling up any given generative models: instead of making generative models larger by stacking more layers, we demonstrated that it was possible to improve the generation quality by turning an image generative model trained on images into its video generative model counterpart trained on pseudo videos, which is usually straightforward since many video generative models are built upon image generative models. On the other hand, this raises challenges on how to design informative pseudo videos. In autoregressive video generation frameworks, we show the potential issue of first-order Markov pseudo videos theoretically and propose to use higher-order Markov pseudo videos instead to address this issue. However, it is in general unclear what the optimal pseudo videos are within such a large design space, which we leave as a future research question. Another interesting future direction is to explore whether the same principle can be applied to other data modalities. While self-supervised signals can be easily obtained using data augmentation for images, it remains unclear whether there are proper ways to inject self-supervised information for other data modalities, such as text or molecules.

Chapter 6

Conclusion and Future Work

This thesis focused on bridging deep sequence models and probabilistic methods to develop new machine learning models that enjoy the advantages from both fields. We have strived to exploit the natural connections between inductive biases in deep sequence models and existing probabilistic methods to guide our model design, enabling the proposed methods to be compatible with existing progresses in the field of deep sequence models. Chapter 3 and Chapter 4 leverage natural connections between popular deep sequence model architectures and Gaussian processes to build GP models tailored to Transformer architectures and online learning problems, respectively. Chapter 5, inspired by the comparison between diffusion models and HVAE, proposes to improve image-generative models by jointly modeling the sequence of the original image and its pseudo video constructed with self-supervised information. Below, we provide a detailed recap of our contributions made in each chapter in Section 6.1. In Section 6.2, we give a critical overview of the limits of the proposed methods and potential future works to fix them.

6.1 Thesis Summary

In Chapter 3, we identified the equivalence between kernel attention and the posterior mean of SVGP: queries, keys, and values in kernel attention module are equivalent to queried input locations, inducing point locations, and variational parameters associated with the posterior mean in SVGP, respectively. Based on this key observation, we proposed sparse Gaussian process attention (SGPA) which places a GP prior for the attention output and quantifies the uncertainty of the attention output with the posterior covariance (constructed based on an additional variational covariance parameters) of the corresponding SVGP. We stack multiple SGPA layers together to build probabilistic Transformers which are essentially deep GPs, and

train SGPA-based Transformers with variational inference. SGPA is tailored to Transformer architectures and quantifies uncertainty directly in the space of attention output rather than through posterior of weights as in weight-space Bayesian neural network methods. For self-attention based Transformers where standard SGPA requires input-dependent variational covariances obtained via amortization similar to the input-dependent keys and values, we further introduce decoupled SGPA which reduces the computational complexity using techniques from decoupled SVGP, where the variational covariance is constructed purely based on another set of global keys shared across all input sequences. SGPA was empirically evaluated on a suite of predictive tasks and it effectively calibrated the uncertainty in Transformer models while keeping the accuracy to be competitively high.

In Chapter 4, we found that by replacing the deterministic signals with a GP prior in the HiPPO (High-order Polynomial Projection Operators) framework, we can naturally interpret the HiPPO time-varying orthogonal projections as inducing variables of an interdomain SVGP, where the basis functions are time-dependent orthogonal polynomials. These interdomain inducing variables adaptively compress the history of a GP while preserving long-term memory. Based on this, we developed online HiPPO-SVGP (OHSVGP) which leverages the long-range memory preservation capability of HiPPO for online learning tasks. OHSVGP bypassed the cumbersome optimization-based online inducing locations updates of standard online SVGP by updating the prior covariance matrices online via the efficient HiPPO-ODE recurrence, bringing an extra layer of computational efficiency to OHSVGP. Empirically, we evaluated OHSVHP on a suite of online and continual learning tasks and we showed that OHSVGP outperforms existing online sparse GP methods, especially in scenarios requiring long-term memory. The efficient streaming capability and preservation of historical information make OHSVGP well-suited for real-world online learning applications demanding both speed and accuracy.

In Chapter 5, we shifted our focus from predictive tasks to generative modeling. In particular, we explored the benefits of sequence or sequential generative modeling. We began with a comparison between two classes of sequential generative models, diffusion models and HVAEs. Our key observation is that diffusion models can be viewed as HVAEs but they impose direct supervision signals, in the form of pseudo videos created by corrupting the original target image, for the sequence of intermediate variables while standard HVAEs leave these intermediate states fully flexible, which might be one of the reasons why diffusion models beat HVAEs. Inspired by this difference, we explored the possibility of improving other types of image generative models by jointly modeling the distribution of the original image and its corresponding pseudo video containing self-supervised information.

Specifically, we extended two popular image generative model frameworks, VQVAE and Improved DDPM, to their video generative model counterparts and trained them on pseudo videos. We empirically showed that this procedure improves the image generation quality with pseudo videos of just a few frames. Moreover, we theoretically analyzed what probabilistic structure over the pseudo videos may be desirable in autoregressive video generation frameworks. This analysis allows us to identify potential issues of certain pseudo videos including those in the form of first-order Markov chains and we proposed a simple and effective approach to bypass the potential issues by constructing higher-order Markov pseudo videos.

6.2 Limitations and Future Research Directions

We have shown the promise of building probabilistic methods tailored to deep sequence models by keeping inductive biases in deep sequence models in mind, and we believe that this vision can inspire more future works that combine deep sequence models and probabilistic modeling, leading to mutually reinforced improvement. We list a few future research directions that may address the limitations of the methods proposed in this thesis for further improvements.

Extending SGPA to decoder-based Transformers. In Chapter 3, we only consider applying SGPA to encoder-based Transformers where the context in the sequence is fixed. However, many sequence modeling tasks require autoregressive prediction based on decoder-based Transformer (Devlin et al., 2019; OpenAI et al., 2024b). For example, in natural language applications, decoder-based Transformer are often used to predict the next token based on existing context consisting of previously generated tokens. Since the context is kept updated, the static GP prior as in Chapter 3 becomes unideal to represent the dynamical uncertainty associated with the changing unknown information beyond the adaptive context. Sequential Bayesian inference which keeps updating the current prior to be the (approximate) posterior at the previous time step is more suitable to model this adaptive context knowledge. To do so with variational inference (Nguyen et al., 2018), we may replace the static ELBO training objective for encoder-based SGPA with the following online ELBO:

$$\begin{aligned}\mathcal{L}_{ELBO}^{(t)} &= \mathbb{E}_{q_t(\boldsymbol{F}^{L,t}|\boldsymbol{F}^{0,t},\{\boldsymbol{k}^{l,h,t}\}_{l=1,h=1}^{L,H})}[\log p(\boldsymbol{Y}^t|\boldsymbol{F}^{L,t})] \\ &-\mathrm{KL}(q_t(\boldsymbol{F}^{L,t}|\boldsymbol{F}^{0,t},\{\boldsymbol{k}^{l,h,t}\}_{l=1,h=1}^{L,H})||q_{t-1}(\boldsymbol{F}^{L,t-1}|\boldsymbol{F}^{0,t-1},\{\boldsymbol{k}^{l,h,t-1}\}_{l=1,h=1}^{L,H})),\end{aligned} \tag{6.1}$$

where q_{t-1} obtained from the previous time step becomes the prior appearing in the KL regularizer term for the current step inference. Notice that both q_t and q_{t-1} are

deep sparse GPs, but with different set of keys, so the KL regularizer in the above online ELBO can be simplified to KL terms evaluated purely based on inducing keys and values again due to the prior conditional matching condition, similar to the online ELBO for OHSVGP (Eq. 4.6). For autoregressive prediction tasks, the inducing keys and values at time t ($\boldsymbol{k}^{l,h,t}$ and $\boldsymbol{v}^{l,h,t}$) are obtained by augmenting the previous inducing keys and values ($\boldsymbol{k}^{l,h,t-1}$ and $\boldsymbol{v}^{l,h,t-1}$) with an additional key and value computed based on the token generated most recently (a procedure known as Key-Value caching).

More efficient SGPA. Compared with standard Transformers, SGPA suffers from higher computational cost due to the additional computation of the posterior covariance. In Chapter 3, we already used decoupled SVGP techniques to construct posterior covariance purely based on a fixed size of global inducing keys shared across all sequence data points. However, in order to scale SGPA up to large Transformers deployed in modern applications such as large language models, further computational cost reduction is indispensable. For example, random Fourier features (RFFs; (Rahimi and Recht, 2007)) can be used to approximate the kernel to accelerate the computation of kernel matrices (Reid et al., 2023). Additionally, the architecture of SGPA may be tweaked to allow better inference schemes. There are equivalent yet cheaper ways to compute posterior for SGPA based on some special class of kernels. For instance, instead of computing the full covariance, posterior sampling from GP based on Markovian kernels can be computed via smoothing in a linear stochastic differential equation (SDE) framework (Hartikainen and Särkkä, 2010).

HiPPO-SGPA. For decoder-based Transformers where the number of key-value pairs is increasing as we generate new tokens, it becomes tricky to construct suitable posterior covariance that balances between the flexibility of modeling the adaptive knowledge in the growing context and computational efficiency. Standard SGPA constructs variational covariance based on the growing number (T) of amortized inducing keys so that its size also keeps growing, which makes it computationally expensive. Decoupled SGPA constructs variational covariance based on the fixed number (M) of static inducing keys globally shared across all sequences, which may not be flexible enough to summarize the global knowledge in the entire dataset of sequences, especially when the number of sequences is huge (e.g., the large corpus to train large language models). Ideally, for each sequence, we want to maintain a fixed number of adaptive inducing key-value pairs that are dynamically updated, and construct fixed-size adaptive variational covariance based on them. HiPPO-SVGP, introduced in Chapter 4, provides a potential option to achieve this goal since HiPPO maintains a fixed size adaptive memory state to capture the dynamical history. Given

a set of queries, we can use the HiPPO recurrence as introduced in Section 4.4 to obtain interdomain attention matrices, $\boldsymbol{K}_{\boldsymbol{f_q u}}$, and with a new query during the autoregressive prediction, the interdomain attention matrices will be updated via another recurrence step to incorporate the information from the new query (Eq. 4.18). The final components for SGPA computation is the fixed-size adaptive variational values and covariance based on this fixed-size interdomain inducing keys. Instead of obtaining them with projection matrices applied over the sequence inputs of the attention module, which is of adaptive size T, as in standard Transformer, one potential solution is to obtain them by applying another RNN over the sequence inputs to dynamically output the required fixed-size variational parameters.

Building probabilistic SSM based on HiPPO-SVGP. HiPPO-SVGP may also be extended to build probabilistic SSM. Standard SSM layers leverage structured recurrence to obtain memory state summarizing the information in the sequence and then project the memory state to obtain the output (Gu et al., 2022; Gu and Dao, 2023). In HiPPO-SVGP, the concept of memory state (or coefficient) is equivalent to the interdomain inducing points, and they are not deterministic but are distributed according to a variational Gaussian distribution $q(\boldsymbol{u})$. Based on this, we may build probabilistic SSM layers enabling uncertainty quantification in SSM models. Still, we need to figure out sensible ways to stack these probabilistic SSM layers together to build deep models for enhanced flexibility. Moreover, although the structure of the recurrence parameters are designed based on fixed HiPPO recurrence matrices, they are trainable parameters in state-of-the-art deep SSMs (Gu et al., 2022; Gu and Dao, 2023). Hence, we may consider tuning the recurrence parameters in HiPPO-SVGP as well. In this case, the trained recurrence parameters will be associated with some implicit function basis and measures whose analytic forms are unknown to us.

Better pseudo video design. As we showed in Chapter 5, better pseudo video design in pseudo video generation framework may improve the image generation quality. In general, since the pseudo-video framework contains image-generative models as special cases (i.e., $T = 1$), intuitively there will be a configuration of hyperparameters (type of data augmentation, architecture etc.) that makes the performance better than image (1-frame) generation. However, as we mentioned in Section 5.6, pseudo-video framework makes the space of hyperparameters much larger, especially the design space of data augmentation for creating pseudo-videos is huge. For different types of models and architectures, the optimal data augmentation is in general not the same (e.g., Gaussian blur data augmentation empirically works better for C-ViVIT but achieves worse results than Gaussian noise for video diffusion in Chapter 5). While we have theoretically shown some intuition that high-

order Markov data augmentation is preferable in autoregressive video generation models, finding optimal data augmentation or practical suggestions of what data augmentation is more suitable for other types of models remains an open research question. Nevertheless, we believe our analysis and empirical experiments have demonstrated a proof of concept that this is a promising framework with a lot of potential to improve generation performance.

References

Albergo, M. S., Boffi, N. M., and Vanden-Eijnden, E. (2023). Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*.

Arbel, J., Pitas, K., Vladimirova, M., and Fortuin, V. (2023). A primer on Bayesian neural networks: Review and debates. *arXiv preprint arXiv:2309.16314*.

Ashman, M., So, J., Tebbutt, W., Fortuin, V., Pearce, M., and Turner, R. E. (2020). Sparse Gaussian process variational autoencoders. *arXiv preprint arXiv:2010.10177*.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. In *arXiv preprint arXiv:1607.06450*.

Bansal, A., Borgnia, E., Chu, H.-M., Li, J., Kazemi, H., Huang, F., Goldblum, M., Geiping, J., and Goldstein, T. (2023). Cold diffusion: Inverting arbitrary image transforms without noise. *Advances in Neural Information Processing Systems*, 36.

Bayes, T. (1763). Lii. an essay towards solving a problem in the doctrine of chances. By the late rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFR S. *Philosophical transactions of the Royal Society of London*, 53:370–418.

Beal, M. J. (2003). *Variational algorithms for approximate Bayesian inference*. PhD thesis, University of London.

Benita, R., Elad, M., and Keshet, J. (2023). Diffar: Denoising diffusion autoregressive model for raw speech waveform generation. In *arXiv preprint arXiv:2310.01381*.

Blair, M., Thompson, J., Henrey, A., and Chen, B. (2013). Skill-Craft1 Master Table Dataset. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5161N.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In *International Conference on Machine Learning*.

Bradshaw, J., Matthews, A. G. d. G., and Ghahramani, Z. (2017). Adversarial examples, uncertainty, and transfer testing robustness in Gaussian process hybrid deep networks. *arXiv preprint arXiv:1707.02476*.

Brooks, S., Gelman, A., Jones, G., and Meng, X. L. (2011). *Handbook of Markov chain Monte Carlo*. CRC press.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Bui, T. D., Nguyen, C. V., and Turner, R. E. (2017). Streaming sparse Gaussian process approximations. In *Advances in Neural Information Processing Systems*.

Bui, T. D. and Turner, R. E. (2014). Tree-structured Gaussian process approximations. In *Advances in Neural Information Processing Systems*.

Burda, Y., Grosse, R., and Salakhutdinov, R. (2015). Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*.

Burt, D. R., Rasmussen, C. E., and van der Wilk, M. (2019). Rates of convergence for sparse variational Gaussian process regression. In *International Conference on Machine Learning (ICML)*.

Campbell, A., Harvey, W., Weilbach, C., De Bortoli, V., Rainforth, T., and Doucet, A. (2023). Trans-dimensional generative modeling via jump diffusion models. In *Advances in Neural Information Processing Systems*.

Casale, F. P., Dalca, A., Saglietti, L., Listgarten, J., and Fusi, N. (2018). Gaussian process prior variational autoencoders. *Advances in neural information processing systems*, 31.

Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge University Press.

Chang, H., Zhang, H., Jiang, L., Liu, C., and Freeman, W. T. (2022). Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11315–11325.

Chang, P. E., Verma, P., John, S., Solin, A., and Khan, M. E. (2023). Memory-based dual Gaussian processes for sequential learning. In *International Conference on Machine Learning*.

Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Luan, D., and Sutskever, I. (2020a). Generative pretraining from pixels. In *International conference on machine learning*, pages 1691–1703. PMLR.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. *Advances in neural information processing systems*, 31.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020b). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Chen, W., Chen, W., Rastrelli, L., and Li, Y. (2025a). Your image is secretly the last frame of a pseudo video. In *Deep Generative Model in Machine Learning: Theory, Principle and Efficacy (DeLTa) Workshop at ICLR*.

Chen, W., Kiyohara, N., Zhu, H. B. H., Curran-Sebastian, J., Bhatt, S., and Li, Y. (2025b). Recurrent memory for online interdomain Gaussian processes. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Chen, W., Klochkov, Y., and Liu, Y. (2024). Post-hoc bias scoring is optimal for fair classification. In *International Conference on Learning Representations (ICLR)*.

Chen, W., Li, B., Zhang, R., and Li, Y. (2025c). Bayesian computation in deep learning. In *arXiv preprint arXiv:2502.18300*.

Chen, W. and Li, Y. (2023). Calibrating transformers via sparse Gaussian processes. In *International Conference on Learning Representations (ICLR)*.

Cheng, C.-A. and Boots, B. (2017). Variational inference for Gaussian process models with linear complexity. In *Advances in Neural Processing Information Systems*.

Child, R. (2021). Very deep VAEs generalize autoregressive models and can outperform them on images. In *International Conference on Learning Representations*.

Cinquin, T., Immer, A., Horn, M., and Fortuin, V. (2021). Pathologies in priors and inference for Bayesian transformers. In *NeurIPS 2021 "I can't believe it's not better" workshop*.

Coker, B., Bruinsma, W. P., Burt, D. R., Pan, W., and Doshi-Velez, F. (2022). Wide mean-field variational Bayesian neural networks ignore the data. In *International Conference on Artificial Intelligence and Statistics*.

Csató, L. and Opper, M. (2002). Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668.

Damianou, A. C. and Lawrence, N. D. (2013). Deep Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*.

Dao, T. and Gu, A. (2024). Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning (ICML)*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., and Gelly, S. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Dutordoir, V., Durrande, N., and Hensman, J. (2020). Sparse Gaussian processes with spherical harmonic features. In *International Conference on Machine Learning (ICML)*.

Duvenaud, D. K., Nickisch, H., and Rasmussen, C. (2011). Additive Gaussian processes. In *Advances in Neural Information Processing Systems*.

Dwivedi, V. P., Joshi, C. K., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. (2020). Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*.

Fan, X. F., Zhang, S., Chen, B., and Zhou, M. (2020). Bayesian attention modules. In *Advances in Neural Information Processing Systems*.

Fang, Y., Wang, W., Xie, B., Sun, Q., Wu, L., Wang, X., Huang, T., Wang, X., and Cao, Y. (2023). Eva: Exploring the limits of masked visual representation learning at scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19358–19369.

Foong, A. Y. K., Burt, D., Li, Y., and Turner, R. (2020). On the expressiveness of approximate inference in Bayesian neural networks. In *Advances in Neural Information Processing Systems*.

Fortuin, V., Baranchuk, D., Rätsch, G., and Mandt, S. (2020). GP-VAE: Deep probabilistic time series imputation. In *International Conference on Artificial Intelligence and Statistics*, pages 1651–1661. PMLR.

Gal, Y. (2016). Uncertainty in deep learning. *PhD dissertation, University of Cambridge*.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*.

Gal, Y. and Turner, R. E. (2015). Improving the Gaussian process sparse spectrum approximation by representing uncertainty in frequency inputs. In *International Conference on Machine Learning (ICML)*.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., March, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35.

Garifolo, J., Lamel, L., Fisher, W., Fiscus, J., Pallett, D., Dahlgren, N., and Zue, V. (1993). TIMIT acoustic-phonetic continuous speech corpus LDC93S1. In *Philadelphia: Linguistic Data Consortium*.

Gat, I., Remez, T., Shaul, N., Kreuk, F., Chen, R. T., Synnaeve, G., Adi, Y., and Lipman, Y. (2024). Discrete flow matching. *arXiv preprint arXiv:2407.15595*.

Gelfand, A. E. (2000). Gibbs sampling. *Journal of the American statistical Association*, 95(452):1300–1304.

Gershman, S. J. and Goodman, N. D. (2014). Amortized inference in probabilistic reasoning. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 36.

Gneiting, T., Balabdaoui, F., and Raftery, A. E. (2007). Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):243–268.

Graves, A., Jaitly, N., and Mohamed, A.-r. (2013a). Hybrid speech recognition with deep bidirectional LSTM. In *2013 IEEE workshop on automatic speech recognition and understanding*, pages 273–278. IEEE.

Graves, A., Mohamed, A.-r., and Hinton, G. E. (2013b). Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*.

Gu, A. and Dao, T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Gu, A., Dao, T., Ermon, S., Rudra, A., and Ré, C. (2020). HiPPO: Recurrent memory with optimal polynomial projections. In *Advances in Neural Information Processing Systems*.

Gu, A., Goel, K., and Ré, C. (2022). Efficiently modeling long sequences with structured state spaces. In *The International Conference on Learning Representations*.

Gu, A., Johnson, I., Timalsina, A., Rudra, A., and Re, C. (2023). How to train your HIPPO: State space models with generalized orthogonal basis projections. In *International Conference on Learning Representations*.

Gulrajani, I., Kumar, K., Ahmed, F., Taiga, A. A., Visin, F., Vazquez, D., and Courville, A. (2017). PixelVAE: A latent variable model for natural images. In *International Conference on Learning Representations*.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. In *International Conference on Machine Learning*.

Hartikainen, J. and Särkkä, S. (2010). Kalman filtering and smoothing solutions to temporal Gaussian process regression models. In *IEEE International Workshop on Machine Learning for signal processing*.

Harvey, W., Naderiparizi, S., Masrani, V., Weilbach, C., and Wood, F. (2022). Flexible diffusion modeling of long video. In *Advances in neural information processing systems*.

Hawryluk, I., Hoeltgebaum, H., Mishra, S., Miscouridou, X., Schnekenberg, R. P., Whittaker, C., Vollmer, M., Flaxman, S., Bhatt, S., and Mellan, T. A. (2021). Gaussian process nowcasting: application to covid-19 mortality reporting. In *Uncertainty in Artificial Intelligence*, pages 1258–1268. PMLR.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022). Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *arXiv:1512.03385*.

Hendrycks, D. and Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*.

Hensman, J., Durrande, N., and Solin, A. (2018). Variational Fourier features for Gaussian processes. *Journal of Machine Learning Research*, 18(151):1–52.

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In *The Conference on Uncertainty in Artificial Intelligence*.

Hensman, J., Matthews, A. G. d. G., and Ghahramani, Z. (2015). Scalable variational Gaussian process classification. In *International Conference on Artificial Intelligence and Statistics*.

Hersbach, H., Bell, B., Berrisford, P., Biavati, G., Horányi, A., Muñoz Sabater, J., Nicolas, J., Peubey, C., Radu, R., Rozum, I., Schepers, D., Simmons, A., Soci, C., Dee, D., and Thépaut, J. N. (2023). ERA5 hourly data on single levels from 1940 to present. Copernicus Climate Change Service (C3S) Climate Data Store (CDS). https://doi.org/10.24381/cds.adbb2d47.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*.

Hoogeboom, E. and Salimans, T. (2023). Blurring diffusion models. In *International Conference on Learning Representations*.

Horn, R. A. and Johnson, C. R. (1991). *Topics in Matrix Analysis*. Cambridge University Press.

Huang, C.-W., Lim, J. H., and Courville, A. C. (2021). A variational perspective on diffusion-based generative models and score matching. *Advances in Neural Information Processing Systems*, 34:22863–22876.

Jafrasteh, B., Villacampa-Calvo, C., and Hernandez-Lobato, D. (2022). Input dependent sparse Gaussian processes. In *International Conference on Machine Learning*.

Jayasekera, I. S., Si, J., Valdettaro, F., Chen, W., Faisal, A. A., and Li, Y. (2025). Variational uncertainty decomposition for in-context learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jazbec, M., Ashman, M., Fortuin, V., Pearce, M., Mandt, S., and Rätsch, G. (2021). Scalable Gaussian process variational autoencoders. In *International Conference on Artificial Intelligence and Statistics*, pages 3511–3519. PMLR.

Johnson, J., Alahi, A., and Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 694–711. Springer.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., et al. (2021). Highly accurate protein structure prediction with alphafold. *Nature*, 596:583–589.

Jung, Y., Lee, H., Chen, W., Möllenhoff, T., Li, Y., Lee, J., and Khan, M. E. (2025a). Compact memory for continual logistic regression. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jung, Y., Lee, H., Chen, W., Möllenhoff, T., Li, Y., Lee, J., and Khan, M. E. (2025b). Compact memory for k-prior based continual learning. In *Symposium on Advances in Approximate Bayesian Inference (AABI)*.

Kapoor, S., Karaletsos, T., and Bui, T. D. (2021). Variational auto-regressive Gaussian processes for continual learning. In *International Conference on Machine Learning*.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2020). Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119.

Khemakhem, I., Kingma, D., Monti, R., and Hyvarinen, A. (2020). Variational autoencoders and nonlinear ICA: A unifying framework. In *International conference on artificial intelligence and statistics*, pages 2207–2217. PMLR.

Kim, J., Nguyen, T. D., Min, S., Cho, S., Lee, M., Lee, H., and Hong, S. (2022). Pure transformers are powerful graph learners. *arXiv*, abs/2207.02505.

Kingma, D., Salimans, T., Poole, B., and Ho, J. (2021). Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational Bayes. In *International Conference on Learning Representations*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. (2017). Self-normalizing neural networks. In *Advances in Neural Information Processing Systems*.

Kristiadi, A., Hein, M., and Hennig, P. (2020). Being Bayesian, even just a bit, fixes overconfidence in relu networks. In *International Conference on Machine Learning*.

Krizhevsky, A., Nair, V., and Hinton, G. (2009). Cifar-10 and CIFAR-100 datasets.

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

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*.

Lean, J. (2004). Solar irradiance reconstruction. In *Data contribution series # 2004-035, IGBP PAGES/World Data Center for Paleoclimatology NOAA/NGDC Paleoclimatology Program, Boulder, CO, USA*.

LeCun, Y.and Bengio, Y. and Hinton, G. (2015). Deep learning. *Nature*, 521:436–444.

Leibfried, F., Dutordoir, V., John, S., and Durrande, N. (2020). A tutorial on sparse Gaussian processes and variational inference. *arXiv preprint arXiv:2012.13962*.

Li, K., Li, X., Wang, Y., He, Y., Wang, Y., Wang, L., and Qiao, Y. (2024). Videomamba: State space model for efficient video understanding. In *arXiv preprint arXiv:2403.06977*.

Li, Y. (2018). Approximate inference: New visions. *PhD dissertation, University of Cambridge*.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. (2023). Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*.

Liu, J. Z., Lin, Z., Padhy, S., Tran, D., Bedrax-Weiss, T., and Lakshminarayanan, B. (2020). Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. In *Advances in Neural Information Processing Systems*.

Liu, X., Gong, C., and Liu, Q. (2022). Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*.

Liu, X., Zhang, F., Hou, Z., Mian, L., Wang, Z., Zhang, J., and Tang, J. (2021). Self-supervised learning: Generative or contrastive. *IEEE transactions on knowledge and data engineering*, 35(1):857–876.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *International Conference on Computer Vision (ICCV)*.

Locatello, F., Bauer, S., Lucic, M., Rätsch, G., Gelly, S., Schölkopf, B., and Bachem, O. (2019). Challenging common assumptions in the unsupervised learning of disentangled representations. In *International Conference on Machine Learning*.

Lou, A., Meng, C., and Ermon, S. (2024). Discrete diffusion modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning (ICML)*.

Lázaro-Gredilla, M. and Figueiras-Vidal, A. (2009). Inter-domain Gaussian processes for sparse inference using inducing features. In *Advances in Neural Information Processing Systems*.

Ma, C. and Hernández-Lobato, J. M. (2021). Functional variational inference based on stochastic process generators. In *Advances in Neural Information Processing Systems*.

Ma, Y., Chen, T., and Fox, E. (2015). A complete recipe for stochastic gradient MCMC. In *Advances in Neural Information Processing Systems*.

Maaløe, L., Fraccaro, M., Liévin, V., and Winther, O. (2019). Biva: A very deep hierarchy of latent variables for generative modeling. *Advances in neural information processing systems*, 32.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D.; Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *NAACL-HLT*.

Maddox, W. J., Stanton, S., and Wilson, A. G. (2021). Conditioning sparse variational Gaussian processes for online decision-making. In *Advances in Neural Information Processing Systems*.

Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure, 405(2): 442–451.*

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133.

Mercer, J. (1909). Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society*, A(209):415–446.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., and Teller, A. H. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics 21.6.* 1087–1092.

Mukhoti, J., Kulharia, V., Sanyal, A., Golodetz, S., Torr, P. H., and Dokania, P. K. (2020). Calibrating deep neural networks using focal loss. In *Advances in Neural Information Processing Systems.*

Naeini, M. P., Cooper, G. F., and Hauskrecht, M. (2015). Obtaining well calibrated probabilities using Bayesian binning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *International Conference on Machine Learning.*

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. (2018). Variational continual learning. In *International Conference on Learning Representations.*

Nichol, A. and Dhariwal, P. (2021). Improved denoising diffusion probabilistic models. *arXiv preprint arXiv:2102.09672.*

OpenAI, :, Hurst, A., Lerer, A., Goucher, A. P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A., Hayes, A., Radford, A., et al. (2024a). Gpt-4o system card. In *arXiv preprint arXiv:2410.21276.*

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., et al. (2024b). Gpt-4 technical report. In *arXiv preprint arXiv:2303.08774.*

Papamarkou, T., Skoularidou, M., Palla, K., Aitchison, L., Arbel, J., Dunson, D., Filippone, M., Fortuin, V., et al. (2024). Position: Bayesian deep learning is needed in the age of large scale ai. *arXiv preprint arXiv:2402.00809.*

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Quan, C. and Li, X. (2024). Multichannel long-term streaming neural speech enhancement for static and moving speakers. In *arXiv preprint arXiv:2403.07675*.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian processes for machine learning*. The MIT Press. ISBN: 0-262-18253-X.

Reid, I., Choromanski, K., Likhosherstov, V., and Weller, A. (2023). Simplex random features. In *International Conference on Machine Learning*.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1278–1286.

Rissanen, S., Heinonen, M., and Solin, A. (2023). Generative modelling with inverse heat dissipation. *International Conference on Learning Representations*.

Ritter, H., Botev, A., and Barber, D. (2018). Online structured laplace approximations for overcoming catastrophic forgetting. In *Advances in Neural Information Processing Systems*, volume 31.

Ritter, H., Kukla, M., Zhang, C., and Li, Y. (2021). Sparse uncertainty representation in deep learning with inducing weights. *Advances in Neural Information Processing Systems*, 34:6515–6528.

Roberts, S., Osborne, M., Ebden, M., Reece, S., Gibson, N., and Aigrain, S. (2013). Gaussian processes for time-series modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 371(1984):20110550*.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65:386.

Rudin, W. (1994). Fourier analysis on groups. *Wiley Classics Library. Wiley-Interscience New York, reprint edition.*

Salimans, T. (2016). A structured variational auto-encoder for learning deep hierarchies of sparse features. *arXiv preprint arXiv:1602.08734*.

Salimans, T., Kingma, D., and Welling, M. (2015). Markov chain Monte Carlo and variational inference: Bridging the gap. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Salimbeni, H., Cheng, C.-A., Boots, B., and Deisenroth, M. (2018). Orthogonally decoupled variational Gaussian processes. In *Advances in Neural Information Processing Systems*.

Salimbeni, H. and Deisenroth, M. (2017). Doubly stochastic variational inference for deep Gaussian processes. In *Advances in Neural Information Processing Systems*.

Särkkä, S. and Solin, A. (2019). *Applied stochastic differential equations*, volume 10. Cambridge University Press.

Shi, Y., De Bortoli, V., Campbell, A., and Doucet, A. (2023). Diffusion schrödinger bridge matching. *Advances in Neural Information Processing Systems*, 36.

Shwartz Ziv, R. and LeCun, Y. (2024). To compress or not to compress—self-supervised learning and information theory: A review. *Entropy*, 26(3):252.

Snelson, E. and Ghahramani, Z. (2005). Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. (2016). Ladder variational autoencoders. *Advances in neural information processing systems*, 29.

Song, J., Meng, C., and Ermon, S. (2021a). Denoising diffusion implicit models. In *International Conference on Learning Representations*.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021b). Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*.

Stanton, S., Maddox, W. J., Delbridge, I., and Wilson, A. G. (2021). Kernel interpolation for scalable online Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*. PMLR.

Sun, S., Shi, J., Wilson, A. G., and Grosse, R. (2021). Scalable variational Gaussian processes via harmonic kernel decomposition. In *International Conference on Machine Learning*.

Sun, S., Zhang, G., Shi, J., and Grosse, R. (2019). Functional variational Bayesian neural networks. In *International Conference on Learning Representation*.

Tfekci, P. and Kaya, H. (2014). Combined Cycle Power Plant. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5002N.

Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. (2020). What makes for good views for contrastive learning? *Advances in neural information processing systems*, 33:6827–6839.

Titsias, M. (2009). Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*.

Ton, J.-F., Flaxman, S., Sejdinovic, D., and Bhatt, S. (2018). Spatial mapping with Gaussian processes and nonstationary fourier features. *Journal of Spatial Statistics*, 28:59–78.

Touvron, H., Cord, M., and Jegou, H. (2022). Deit iii: Revenge of the vit. *arXiv preprint arXiv:2204.07118*.

Tran, D., Dusenberry, M. W., van der Wilk, M., and Hafner, D. (2019). Bayesian layers: A module for neural network uncertainty. *arXiv:1812.03973*.

Tran, G.-L., Milios, D., Michiardi, P., and Filippone, M. (2021). Sparse within sparse Gaussian processes using neighbor information. In *International Conference on Machine Learning*.

Tsai, Y.-H. H., Bai, S., Yamada, M., Morency, L.-P., and Salakhutdinov, R. (2019). Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. In *EMNLP*.

Vahdat, A. and Kautz, J. (2020). Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 33:19667–19679.

van den Oord, A., Kalchbrenner, N., Espeholt, L., kavukcuoglu, K., Vinyals, O., and Graves, A. (2016a). Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*.

van den Oord, A., Kalchbrenner, N., and kavukcuoglu, K. (2016b). Pixel recurrent neural networks. In *International Conference on Machine Learning*.

Van Den Oord, A., Vinyals, O., et al. (2017). Neural discrete representation learning. *Advances in neural information processing systems*, 30.

Van der Wilk, M., Dutordoir, V., John, S., Artemev, A., Adam, V., and Hensman, J. (2020). A framework for interdomain and multioutput Gaussian processes. *arXiv preprint arXiv:2003.01115*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Villegas, R., Babaeizadeh, M., Kindermans, P.-J., Moraldo, H., Zhang, H., Saffar, M. T., Castro, S., Kunze, J., and Erhan, D. (2022). Phenaki: Variable length video generation from open domain textual descriptions. In *International Conference on Learning Representations*.

Wang, F.-Y., Yang, L., Huang, Z., Wang, M., and Li, H. (2024). Rectified diffusion: Straightness is not your need in rectified flow. *arXiv preprint arXiv:2410.07303*.

Warstadt, A., Singh, A., and Bowman, S. R. (2019). Neural network acceptability judgments. In *Transactions of the Association for Computational Linguistics*.

Wen, B., Xu, C., Wolfe, R., Wang, L. L., and Howe, B. (2024). Mitigating overconfidence in large language models: A behavioral lens on confidence estimation and calibration. In *NeurIPS 2024 Workshop on Behavioral Machine Learning*.

Wilkinson, W., Solin, A., and Adam, V. (2021). Sparse algorithms for Markovian Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 1747–1755. PMLR.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. (2016). Deep kernel learning. In *International Conference on Artificial Intelligence and Statistics*.

Wilson, A. G. and Izmailov, P. (2022). Bayesian deep learning and a probabilistic perspective of generalization. *arXiv preprint arXiv:2002.08791*.

Wilson, J. T., Borovitskiy, V., Terenin, A., Mostowsky, P., and Deisenroth, M. P. (2020). Efficiently sampling functions from Gaussian process posteriors. In *International Conference on Machine Learning*.

Wu, M., Zhuang, C., Mosse, M., Yamins, D., and Goodman, N. (2020). On mutual information in contrastive learning for visual representations. *arXiv preprint arXiv:2005.13149*.

Xiao, T. Z. and Bamler, R. (2023). Trading information between latents in hierarchical variational autoencoders. In *The Eleventh International Conference on Learning Representations*.

Xiong, M., Hu, Z., Lu, X., Li, Y., Fu, J., He, J., and Hooi, B. (2024). Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs. In *International Conference on Learning Representations (ICLR)*.

Xue, B., Yu, J., Xu, J., Liu, S., Hu, S., Ye, Z., Geng, M., Liu, X., and Meng, H. (2021). Bayesian transformer language models for speech recognition. *arXiv:2102.04754*.

Yan, W., Zhang, Y., Abbeel, P., and Srinivas, A. (2021). Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*.

Zeni, C., Pinsler, R., Zügner, D., Fowler, A., Horton, M., Fu, X., Shysheya, S., Crabbé, J., Sun, L., Smith, J., Nguyen, B., Schulz, H., Lewis, S., Huang, C.-W., Lu, Z., Zhou, Y., Yang, H., Hao, H., Li, J., Tomioka, R., and Xie, T. (2024). Mattergen: a generative model for inorganic materials design. In *arXiv preprint arXiv:2312.03687*.

Zhang, C., Bütepage, J., Kjellström, H., and Mandt, S. (2018a). Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018b). The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595.

Zhang, R., Li, C., Zhang, J., Chen, C., and Wilson, A. G. (2019). Cyclical stochastic gradient MCMC for Bayesian deep learning. In *International Conference on Learning Representations*.

Zhu, H., Rodas, C. B., and Li, Y. (2023). Markovian Gaussian process variational autoencoders. In *International Conference on Machine Learning*.

Zhu, L., Liao, B., Zhang, Q., Wang, X., Liu, W., and Wang, X. (2024). Vision Mamba: Efficient visual representation learning with bidirectional state space model. In *International Conference on Machine Learning (ICML)*.

Appendix A

Supplementary Material for Chapter 2

A.1 HiPPO Variants in Addition to HiPPO-LegS

In addition to HiPPO-LegS, we also provide here the formulations of a few other HiPPO variants (Gu et al., 2020) based on non-uniform measures.

HiPPO-LegT. HiPPO-LegT uses a uniform measure over the most recent context window with length θ, $\omega^{(t)}(x) = \frac{1}{\theta}\mathbf{1}_{[t-\theta,t]}(x)$ and scaled Legendre polynomials with input domain adapted to $[t-\theta, t]$, $g_m^{(t)}(x) = (2m+1)^{1/2} P_m\left(\frac{2(x-t)}{\theta} + 1\right)$, as basis functions, where $P_m(\cdot)$ is the standard m-th Legendre polynomial with input domain $[-1, 1]$. The formulas of matrices $\boldsymbol{A}(t) \in \mathbb{R}^{M\times M}$ and $\boldsymbol{B}(t) \in \mathbb{R}^{M\times 1}$ for the corresponding linear ODE are:

$$\boldsymbol{A}(t) = -\frac{1}{\theta}\boldsymbol{A}, \quad [\boldsymbol{A}]_{mk} = \begin{cases} \sqrt{(2m+1)(2k+1)} & \text{if } m \geq k \\ (-1)^{m-k}\sqrt{(2m+1)(2k+1)} & \text{if } m < k \end{cases} \tag{A.1}$$

and

$$[\boldsymbol{B}(t)]_m = \frac{[\boldsymbol{B}]_m}{\theta} = \frac{\sqrt{2m+1}}{\theta}, \tag{A.2}$$

HiPPO-LagT. HiPPO-LagT uses an exponential decay measure, which assigns more importance to more recent history, $\omega^{(t)}(x) = \exp\left(-(t-x)\right)\mathbf{1}_{(-\infty,t]}(x)$ and scaled Laguerre polynomials with input domain adapted to $(-\infty, t]$, $g_m^{(t)}(x) = L_m\left(t-x\right)$, as basis functions, where $L_m(\cdot)$ is the standard m-th Laguerre polynomial with input domain $[0, \infty)$. The formulas of matrices $\boldsymbol{A}(t) \in \mathbb{R}^{M\times M}$ and

$\boldsymbol{B}(t) \in \mathbb{R}^{M\times 1}$ for the corresponding linear ODE are:

$$[\boldsymbol{A}(t)]_{mk} = [\boldsymbol{A}]_{mk} = \begin{cases} -1 & \text{if } m \geq k \\ 0 & \text{if } m < k \end{cases} \tag{A.3}$$

and

$$[\boldsymbol{B}(t)]_m = [\boldsymbol{B}]_m = 1, \tag{A.4}$$

HiPPO-FouT. Similar to HiPPO-LegT, HiPPO-FouT also uses the uniform measure over a sliding window window, $\omega^{(t)}(x) = \frac{1}{\theta}\mathbf{1}_{[t-\theta,t]}(x)$, but is based on scaled Fourier basis functions with input domain adapted to $[t-\theta, t]$, $g_m^{(t)}(x) = \exp\left(2\pi i m \frac{t-x}{\theta}\right)$, as basis functions. The formulas of matrices $\boldsymbol{A}(t) \in \mathbb{R}^{M\times M}$ and $\boldsymbol{B}(t) \in \mathbb{R}^{M\times 1}$ for the corresponding linear ODE are:

$$[\boldsymbol{A}(t)]_{mk} = [\boldsymbol{A}]_{mk} = \begin{cases} -\frac{1}{\theta} & \text{if } m \neq k \\ \frac{2\pi i m - 1}{\theta} & \text{if } m = k \end{cases} \tag{A.5}$$

and

$$[\boldsymbol{B}(t)]_m = [\boldsymbol{B}]_m = \frac{1}{\theta}, \tag{A.6}$$

Appendix B

Supplementary Material for Chapter 3

B.1 Derivations

B.1.1 ELBO Derivations for SVGP

For completeness, we include the full derivation of ELBO as shown in Eq. 2.38 for standard SVGP (Titsias, 2009; Hensman et al., 2013). With M inducing points pairs $\{(\boldsymbol{z}_m, u_m)\}_{m=1}^M$, the prior distribution of $[\boldsymbol{f}, \boldsymbol{u}]^\top$ is:

$$p(\boldsymbol{f}, \boldsymbol{u}|\boldsymbol{X}, \boldsymbol{Z}) = \mathcal{N}(\mathbf{0}, \begin{pmatrix} \boldsymbol{K}_{\boldsymbol{XX}} & \boldsymbol{K}_{\boldsymbol{XZ}} \\ \boldsymbol{K}_{\boldsymbol{ZX}} & \boldsymbol{K}_{\boldsymbol{ZZ}} \end{pmatrix}). \tag{B.1}$$

With prior conditional matching assumption (see Eq. 2.35), the approximate posterior conditional distribution of function values $\boldsymbol{f}$ for inputs $\boldsymbol{X}$ given inducing points $\boldsymbol{u}$ is the same as the prior conditional distribution:

$$q(\boldsymbol{f}|\boldsymbol{u}, \boldsymbol{Z}, \boldsymbol{X}) = p(\boldsymbol{f}|\boldsymbol{u}, \boldsymbol{Z}, \boldsymbol{X}). \tag{B.2}$$

Under prior conditional matching assumption, $q(\boldsymbol{f}, \boldsymbol{u}|\boldsymbol{Z}, \boldsymbol{X}) = p(\boldsymbol{f}|\boldsymbol{u}, \boldsymbol{Z}, \boldsymbol{X})q(\boldsymbol{u})$, where $q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{m}_{\boldsymbol{u}}, \boldsymbol{S}_{\boldsymbol{u}})$. Suppose the observation likelihood is $p(\boldsymbol{y}|\boldsymbol{f})$, ELBO can

be simplified as follows:

$$\begin{aligned}
\mathcal{L}_{ELBO} &= \mathbb{E}_{q(\boldsymbol{f},\boldsymbol{u}|\boldsymbol{Z},\boldsymbol{X})}[\log \frac{p(\boldsymbol{y},\boldsymbol{f},\boldsymbol{u}|\boldsymbol{Z},\boldsymbol{X})}{q(\boldsymbol{f},\boldsymbol{u}|\boldsymbol{Z},\boldsymbol{X})}] \\
&= \mathbb{E}_{q(\boldsymbol{f},\boldsymbol{u}|\boldsymbol{Z},\boldsymbol{X})}[\log \frac{p(\boldsymbol{y}|\boldsymbol{f})\cancel{p(\boldsymbol{f}|\boldsymbol{u},\boldsymbol{Z},\boldsymbol{X})}p(\boldsymbol{u}|\boldsymbol{Z})}{\cancel{p(\boldsymbol{f}|\boldsymbol{u},\boldsymbol{Z},\boldsymbol{X})}q(\boldsymbol{u})}] \\
&= \int (\int p(\boldsymbol{f}|\boldsymbol{u},\boldsymbol{Z},\boldsymbol{X})q(\boldsymbol{u})d\boldsymbol{u}) \log p(\boldsymbol{y}|\boldsymbol{f})d\boldsymbol{f} \\
&\qquad + \int q(\boldsymbol{u}) \log \frac{p(\boldsymbol{u}|\boldsymbol{Z})}{q(\boldsymbol{u})} \cancel{\int p(\boldsymbol{f}|\boldsymbol{u},\boldsymbol{Z},\boldsymbol{X})d\boldsymbol{f}}d\boldsymbol{u} \\
&= \underbrace{\mathbb{E}_{q(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{Z})}[\log p(\boldsymbol{y}|\boldsymbol{f})]}_{\text{ELL}} - \underbrace{\mathrm{KL}(q(\boldsymbol{u})||p(\boldsymbol{u}|\boldsymbol{Z}))}_{\text{KL regularizer}}.
\end{aligned} \tag{B.3}$$

Here $q(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{Z}) = \int p(\boldsymbol{f}|\boldsymbol{u},\boldsymbol{Z},\boldsymbol{X})q(\boldsymbol{u})d\boldsymbol{u}$ is a Gaussian and is given as:

$$q(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{Z}) = \mathcal{N}(\boldsymbol{K}_{\boldsymbol{XZ}}\boldsymbol{K}_{\boldsymbol{ZZ}}^{-1}\boldsymbol{m}_{\boldsymbol{u}}, \boldsymbol{K}_{\boldsymbol{XX}} + \boldsymbol{K}_{\boldsymbol{XZ}}\boldsymbol{K}_{\boldsymbol{ZZ}}^{-1}(\boldsymbol{S}_{\boldsymbol{u}} - \boldsymbol{K}_{\boldsymbol{ZZ}})\boldsymbol{K}_{\boldsymbol{ZZ}}^{-1}\boldsymbol{K}_{\boldsymbol{ZX}}). \tag{B.4}$$

With Gaussian likelihood, the first term in ELBO can be evaluated analytically. Otherwise we estimate the first term using Monte-Carlo samples $\boldsymbol{f} \sim q(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{Z})$. The second term is a KL-divergence between two Gaussian distributions. Thus, it admits a closed form:

$$\mathrm{KL}(q(\boldsymbol{u})||p(\boldsymbol{u}|\boldsymbol{Z})) = \frac{1}{2}\left[Tr(\boldsymbol{K}_{\boldsymbol{ZZ}}^{-1}\boldsymbol{S}_{\boldsymbol{u}}) + \boldsymbol{m}_{\boldsymbol{u}}^{\top}\boldsymbol{K}_{\boldsymbol{ZZ}}^{-1}\boldsymbol{m}_{\boldsymbol{u}} + \log\frac{|\boldsymbol{K}_{\boldsymbol{ZZ}}|}{|\boldsymbol{S}_{\boldsymbol{u}}|}\right] + \text{const.} \tag{B.5}$$

In standard SGPA the ELBO objective remains almost the same, except that as the variational mean is reparameterized to $\boldsymbol{v} := \boldsymbol{K}_{\boldsymbol{ZZ}}^{-1}\boldsymbol{m}_{\boldsymbol{u}}$, the mean of $q(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{Z})$ becomes $\boldsymbol{K}_{\boldsymbol{XZ}}\boldsymbol{v}$, and the quadratic term in $\mathrm{KL}(q(\boldsymbol{u})||p(\boldsymbol{u}|\boldsymbol{Z}))$ becomes $\boldsymbol{v}^{\top}\boldsymbol{K}_{\boldsymbol{ZZ}}\boldsymbol{v}$.

B.1.2 Derivation of ELBO for Transformers Based on SGPA

An L-layer Transformer based on SGPA is a deep GP (Damianou and Lawrence, 2013), and we train it using the doubly stochastic variational inference framework (Salimbeni and Deisenroth, 2017). For each input sequence $\boldsymbol{F}^0 := \boldsymbol{X}$, the joint distribution for $\boldsymbol{Y}, \{\boldsymbol{F}^l\}_{l=1}^L, \{\boldsymbol{u}_{a\cup g}^{l,h}\}_{l=1,h=1}^{L,H}$ is:

$$\begin{aligned}
&p(\boldsymbol{Y},\{\boldsymbol{F}^l\}_{l=1}^L, \{\boldsymbol{u}_{a\cup g}^{l,h}\}_{l=1,h=1}^{L,H}|\boldsymbol{F}^0) = \\
&\qquad p(\boldsymbol{Y}|\boldsymbol{F}^L)[\prod_{l=1}^{L} p(\boldsymbol{F}^l|\{\boldsymbol{u}_{a\cup g}^{l,h}\}_{h=1}^H, \boldsymbol{F}^{l-1})p(\{\boldsymbol{u}_{a\cup g}^{l,h}\}_{h=1}^H|\{\boldsymbol{k}_g^{l,h}\}_{h=1}^H, \boldsymbol{F}^{l-1})],
\end{aligned} \tag{B.6}$$

where $p(\{\boldsymbol{u}_{a\cup g}^{l,h}\}_{h=1}^H|\{\boldsymbol{k}_g^{l,h}\}_{h=1}^H, \boldsymbol{F}^{l-1}) = \prod_{h=1}^H p(\boldsymbol{u}_{a\cup g}^{l,h}|\boldsymbol{k}_g^{l,h}, \boldsymbol{F}^{l-1})$ since we assume the prior for inducing points factorizes across heads in each layer. Note here the amortized keys $\boldsymbol{k}_a^{l,h}$ depend on $\boldsymbol{F}^{l-1}$ in a deterministic manner, therefore we drop the amortized

key terms in the conditioning. Assuming prior conditional matching, i.e,

$$q(\boldsymbol{F}^l|\{\boldsymbol{u}_{a\cup g}^{l,h}\}_{h=1}^H,\boldsymbol{F}^{l-1}) = p(\boldsymbol{F}^l|\{\boldsymbol{u}_{a\cup g}^{l,h}\}_{h=1}^H,\boldsymbol{F}^{l-1}), \tag{B.7}$$

the joint approximate posterior for $\{\boldsymbol{F}^l\}_{l=1}^L, \{\boldsymbol{u}_{a\cup g}^{l,h}\}_{l=1,h=1}^{L,H}$ is:

$$q(\{\boldsymbol{F}^l\}_{l=1}^L, \{\boldsymbol{u}_{a\cup g}^{l,h}\}_{l=1,h=1}^{L,H}|\boldsymbol{F}^0) = \prod_{l=1}^L p(\boldsymbol{F}^l|\{\boldsymbol{u}_{a\cup g}^{l,h}\}_{h=1}^H,\boldsymbol{F}^{l-1}) q(\{\boldsymbol{u}_{a\cup g}^{l,h}\}_{h=1}^H|\{\boldsymbol{k}_g^{l,h}\}_{h=1}^H,\boldsymbol{F}^{l-1}), \tag{B.8}$$

where $q(\{\boldsymbol{u}_{a\cup g}^{l,h}\}_{h=1}^H|\{\boldsymbol{k}_g^{l,h}\}_{h=1}^H,\boldsymbol{F}^{l-1}) = \prod_{h=1}^H q(\boldsymbol{u}_{a\cup g}^{l,h}|\boldsymbol{k}_g^{l,h},\boldsymbol{F}^{l-1})$ since we let the approximate distribution for $\{\boldsymbol{u}_{a\cup g}^{l,h}\}_{h=1}^H$ also factorizes across heads. The ELBO is derived in a similar manner as the single-layer GP case (Eq. B.3) and again the conditional distribution terms in q and p cancel with each other. This simplifies the ELBO to:

$$\begin{aligned}
\mathcal{L}_{ELBO} &= \mathbb{E}_{q(\{\boldsymbol{F}^l\}_{l=1}^L,\{\boldsymbol{u}_{a\cup g}^{l,h}\}_{l=1,h=1}^{L,H}|\boldsymbol{F}^0)}[\log \frac{p(\boldsymbol{Y},\{\boldsymbol{F}^l\}_{l=1}^L,\{\boldsymbol{u}_{a\cup g}^{l,h}\}_{l=1,h=1}^{L,H}|\boldsymbol{F}^0)}{q(\{\boldsymbol{F}^l\}_{l=1}^L,\{\boldsymbol{u}_{a\cup g}^{l,h}\}_{l=1,h=1}^{L,H}|\boldsymbol{F}^0)}] \\
&= \mathbb{E}_{q(\boldsymbol{F}^L|\boldsymbol{F}^0,\{\boldsymbol{k}_g^{l,h}\}_{l=1,h=1}^{L,H})}[\log p(\boldsymbol{Y}|\boldsymbol{F}^L)] \\
&\quad - \sum_{l=1}^L\sum_{h=1}^H \mathbb{E}_{q(\boldsymbol{F}^l|\boldsymbol{F}^0,\{\boldsymbol{k}_g^{j,h}\}_{j=1,h=1}^{l,H}))}\{\mathrm{KL}(q(\boldsymbol{u}_{a\cup g}^{l,h}|\boldsymbol{k}_g^{l,h},\boldsymbol{F}^{l-1})||p(\boldsymbol{u}_{a\cup g}^{l,h}|\boldsymbol{k}_g^{l,h},\boldsymbol{F}^{l-1}))\},
\end{aligned} \tag{B.9}$$

where

$$\begin{aligned}
&q(\boldsymbol{F}^l|\boldsymbol{F}^0,\{\boldsymbol{k}_g^{j,h}\}_{j=1,h=1}^{l,H}) = \\
&\qquad \int \prod_{j=1}^l p(\boldsymbol{F}^j|\{\boldsymbol{u}_{a\cup g}^{j,h}\}_{h=1}^H,\boldsymbol{F}^{j-1}) q(\{\boldsymbol{u}_{a\cup g}^{j,h}\}_{h=1}^H|\{\boldsymbol{k}_g^{j,h}\}_{h=1}^H,\boldsymbol{F}^{j-1}) d\boldsymbol{u}_{a\cup g}^{1:l,1:H} d\boldsymbol{F}^{1:l-1}.
\end{aligned} \tag{B.10}$$

Both terms in the ELBO can be estimated using samples generated iteratively through each layer using the reparameterization trick. For the second "regularization" term, the KL-divergence within the expectation admits a simplified form as we assume for each attention output dimension (d), an independent decoupled SVGP is fitted:

$$\begin{aligned}
&\mathrm{KL}(q(\boldsymbol{u}_{a\cup g}^{l,h}|\boldsymbol{k}_g^{l,h},\boldsymbol{F}^{l-1})||p(\boldsymbol{u}_{a\cup g}^{l,h}|\boldsymbol{k}_g^{l,h},\boldsymbol{F}^{l-1})) \\
&= \frac{1}{2}\sum_{d=1}^D \{[\boldsymbol{v}_a^{l,h}]_{:,d}^\top(\boldsymbol{K}_{\boldsymbol{k}_a^{l,h}\boldsymbol{k}_a^{l,h}} - \boldsymbol{K}_{\boldsymbol{k}_a^{l,h}\boldsymbol{k}_g^{l,h}}\boldsymbol{K}_{\boldsymbol{k}_g^{l,h}\boldsymbol{k}_g^{l,h}}^{-1}\boldsymbol{K}_{\boldsymbol{k}_g^{l,h}\boldsymbol{k}_a^{l,h}})[\boldsymbol{v}_a^{l,h}]_{:,d} + [\boldsymbol{v}_g^{l,h}]_{:,d}^\top\boldsymbol{K}_{\boldsymbol{k}_g^{l,h}\boldsymbol{k}_g^{l,h}}[\boldsymbol{v}_g^{l,h}]_{:,d} \\
&\quad + \mathrm{Tr}([\boldsymbol{S}_g^{l,h}]_{:,:,d}\boldsymbol{K}_{\boldsymbol{k}_g^{l,h}\boldsymbol{k}_g^{l,h}}^{-1}) - \log\det[\boldsymbol{S}_g^{l,h}]_{:,:,d} + \log\det\boldsymbol{K}_{\boldsymbol{k}_g^{l,h}\boldsymbol{k}_g^{l,h}}\} + \mathrm{const},
\end{aligned} \tag{B.11}$$

where D is the total number of attention output dimensions and M_g is the number of global inducing points for each head.

B.2 Uncertainty Calibration Metrics

We use two types of widely used metrics to assess the quality of uncertainty estimates for in-distribution data, proper scoring rule and calibration error. Specifically, negative log-likelihood is a proper scoring rule based metric, and expected calibration error (ECE) and maximum calibration error (MCE) are two calibration error based metrics.

B.2.1 Proper Scoring Rule

A proper scoring rule (Gneiting et al., 2007) is a function taking a distribution and a data point as inputs. Suppose the data generating distribution is p_d, and the predictive distribution provided by our model is $\hat{p}$, then for each observation $\boldsymbol{x}_i$ we can obtain a score, $S(\hat{p}, \boldsymbol{x}_i)$, based on the predictive distribution. To measure the deviation of $\hat{p}$ from p_d, we take the expectation of the aforementioned scores over p_d: $S(\hat{p}, p_d) = \mathbb{E}_{p_d}[S(\hat{p}, \boldsymbol{x})]$. For a proper scoring rule, $S(p_d, p_d) \leq S(\hat{p}, p_d), \quad \forall \hat{p}$, so the minimum of $S(\hat{p}, p_d)$ can be achieved when $\hat{p} = p_d$. For a scoring rule which is strictly proper, $S(p_d, p_d)$ is the only global minimum. This justifies why it is proper for assessing the quality of predictive distribution. In practice, $S(\hat{p}, p_d)$ is analytically intractable since the ground-truth p_d is unknown, and we typically resort to Monte-Carlo to estimation. NLL is one popular metric computed based on a proper scoring rule, specifically the logarithm score. Given a test set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{n=1}^N$, NLL is computed via:

$$\text{NLL} = -\mathbb{E}_{p_d}[\log \hat{p}(\boldsymbol{y}|\boldsymbol{x})] \approx -\frac{1}{N}\sum_{n=1}^{N} \log \hat{p}(\boldsymbol{y}_n \mid \boldsymbol{x}_n).$$

B.2.2 Calibration Error

Calibration error (Naeini et al., 2015; Guo et al., 2017) measures the deviation between the model confidence in classification correctness and the ground truth probability of correctness. The predicted probability, $\hat{p}$, for the class the data point is classified to reflects the model's confidence in the classification correctness. For a well-calibrated model, for each confidence level $\hat{p}$, the proportion p of the data points belonging to this "$\hat{p}$-confidence group" that are classified correctly should be equal to $\hat{p}$. Calibration error is then computed based on the absolute difference between $\hat{p}$ and p. The ECE is the expectation of this difference w.r.t. the distribution of

confidence $\hat{p}$:

$$ECE = \mathbb{E}_{\hat{p}}\left[|\hat{p} - p|\right] \tag{B.12}$$

The MCE is the maximum of these differences:

$$MCE = \arg\max_{\hat{p}} |\hat{p} - p| \tag{B.13}$$

In practice, since we do not have sufficient amount of data for each confidence level, a biased estimation of calibration errors is obtained using histogram binning (Naeini et al., 2015; Guo et al., 2017). We partition the domain of $\hat{p}$, $[0, 1]$, into 15 bins of equal size as in Guo et al. (2017).

B.3 Additional Experiments

B.3.1 Empirical Comparison Between Standard SGPA and Decoupled SGPA

In our preliminary experiments, we compare performance of ViT based on standard SGPA versus decoupled SGPA for image classification on CIFAR10 without data augmentation. We also consider decoupled SGPA based on DSVGP (Cheng and Boots, 2017). According to Eq. 3.10, the posterior mean and covariance formula for decoupled SGPA based on DSVGP (Cheng and Boots, 2017) is given as follows:

$$\begin{aligned} \boldsymbol{m}_d &= \boldsymbol{K}_{\boldsymbol{q}\boldsymbol{k}_a}[\boldsymbol{v}_a]_{:,d}, \\ \boldsymbol{\Sigma}_d &= \boldsymbol{K}_{\boldsymbol{q}\boldsymbol{q}} + \boldsymbol{K}_{\boldsymbol{q}\boldsymbol{k}_g}\boldsymbol{K}_{\boldsymbol{k}_g\boldsymbol{k}_g}^{-1}([\boldsymbol{S}_g]_{:,:,d} - \boldsymbol{K}_{\boldsymbol{k}_g\boldsymbol{k}_g})\boldsymbol{K}_{\boldsymbol{k}_g\boldsymbol{k}_g}^{-1}\boldsymbol{K}_{\boldsymbol{k}_g\boldsymbol{q}}, \end{aligned} \tag{B.14}$$

Table B.1 shows ViT based on standard SGPA and decoupled SGPA based on DSVGP (Cheng and Boots, 2017) achieve worse performance than decouple SGPA based on ODSVGP (Salimbeni et al., 2018). In particular, standard SGPA considerably underfits the data. Therefore we only consider decoupled SGPA based on ODSVGP (Salimbeni et al., 2018) for the rest of the experiments.

Table B.1: Test set accuracy and NLL of ViTs based on standard and two variants of decoupled SGPA, trained on CIFAR10 without data augmentation.

Model	Accuracy	NLL
Standard SGPA	0.6435±0.0039	1.0159±0.0065
Decoupled SGPA (Cheng and Boots, 2017)	0.7513±0.0014	0.7877±0.0045
Decoupled SGPA (Salimbeni et al., 2018)	**0.7787±0.0024**	**0.6968±0.0032**

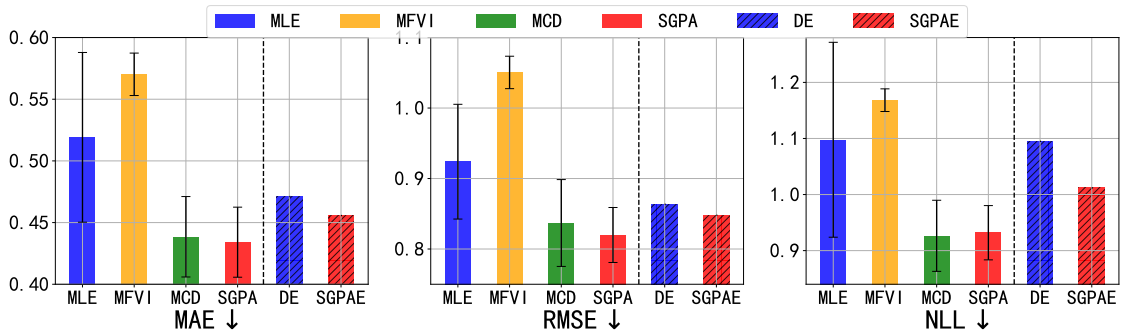


Figure B.1: Test set regression error and NLL metrics for Transformers trained on ZINC.

B.3.2 Graph Property Regression with ZINC Dataset

For graph property regression, we assume a Laplace likelihood with a trainable scale parameter b (i.e., the density of the observation likelihood is $g(y|f) = \frac{1}{2b}\exp(-\frac{|y-f|}{b})$, where f is the scalar function value output by the Transformer). We compute mean-absolute-error (MAE), root-mean-square error (RMSE) and negative-log-likelihood (NLL) to evaluate the models, with results presented in Figure B.1. Moreover, we use predictive variances as scores and evaluate OOD detection performance in Figure B.2. Note that MLE is useless for OOD detection in this case since it produces homogeneous predictive variances for all instances. We use a synthetic OOD dataset generated from test set: for each test instance, we remove the existing edges from the adjacency matrix and add edges between nodes that are not originally connected. Within "single-model" methods, SGPA and MCD achieve much better results than MLE and MFVI. For this task, the difference in performance between SGPA and MCD is negligible. However, when compared to MCD, SGPA performs more robustly as it returns smaller error bars. Ensemble methods outperform

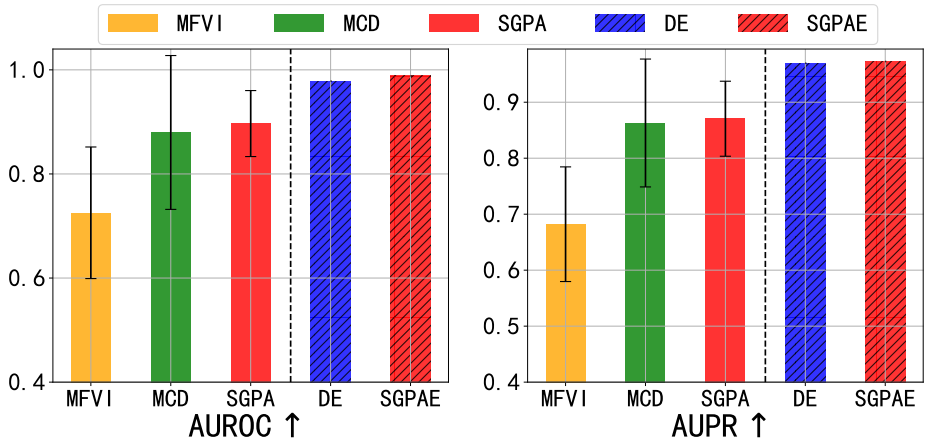


Figure B.2: AUROC and AUPR metrics for OOD detection using Transformers trained on ZINC.

"single-model" methods in OOD detection with SGPAE achieving the best result. Interestingly, for in-distribution calibration, they achieve worse performance than SGPA and MCE.

B.4 Detailed Experimental Setups and Hyperparameters

Training settings shared across experiments. For MLE and MCD, we initially considered both dropout rates 0.1 and 0.2, but we found models trained with dropout rate 0.1 consistently outperformed models trained with dropout rate 0.2 in terms of accuracy in our preliminary experiments for image classification. Therefore, we decided to use dropout rate 0.1 for all methods except MFVI. For each layer, we use mean-pooling strategy. The non-linear mapping G_{ϕ^l} at each attention layer is parameterized by a 2-layer MLP as in Vaswani et al. (2017). For models with kernel-based attentions, we use exponential kernel for sentiment analysis and linguistic acceptability, (Tsai et al., 2019): $k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp(\sum_{j=1}^{D} \frac{x_j x_j'}{\sigma_j^2})$, and we use ARD-RBF kernel (Rasmussen and Williams, 2006) for image classification and graph property regression: $k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp(-\frac{1}{2} \sum_{j=1}^{D} \frac{(x_j - x_j')^2}{\sigma_j^2})$, where D is the dimension of $\boldsymbol{x}$ and $\boldsymbol{x}'$, σ_f^2 is the output variance, and σ_j is the length-scale for the j-th dimension. For MFVI, MCD, SNGP and SGPA, predictive uncertainty is estimated using 10 Monte Carlo samples.

- **Initialization:** we train all our models from scratch (i.e. all parameters are randomly initialized without any pretraining). Apart from the global inducing points parameters, the rest of the parameters are the same as in standard transformers, and are initialized via the default method of the deep learning platform (we use Pytorch (Paszke et al., 2019)). Each dimension of global inducing locations and the global variational mean are randomly initialized with standard Gaussian. For the Cholesky factor used to parameterize the global variational covariance, we randomly initialize each element in the lower triangular part and each element in the log-diagonal with a standard Gaussian.

- **Optimization:** all the models are trained using ADAM optimizer (Kingma and Ba, 2015), and for each input sequence in a batch, we only draw one sample to estimate the ELBO (eq. 3.18). In our experiments, we observe no optimization issue: the ELBO is consistently minimized during the training without significant spikes.

Sentiment analysis with IMDB (Maas et al., 2011). We consider 5 different splits, each includes 35,000 training, 5,000 validation, and 10,000 test instances. The

maximum number of tokens in each input sequence is 512. Architecture-wise, we use a Transformer with 1 MHSA layer and 8 attention heads, same embedding dimension and hidden dimension of 128. For SGPA we use 50 global inducing points for each head. We train all the models (except post-hoc methods, TS and KFLLLA) for 20 epochs with batch size 32 and with a initial learning rate 0.001 which decays linearly to 0.0001. The best model is selected based on the validation accuracy computed after every training epoch.

Linguistic acceptability with CoLA (Warstadt et al., 2019). The 516 OOD samples provided by the original dataset are used to assess the models' OOD robustness. Within each of the 5 independent runs, the remaining 9,078 in-distribution samples are randomly split into 7,262 training and 1,816 in-distribution test instances. We use a Transformer with 2 MHSA layers, each with 4 attention heads, an embedding dimension of 128 and a hidden dimension of 256. For the input embeddings, we use ELMO-style representation (Peters et al., 2018). For SGPA we use 5 global inducing points for each head. We train all models (except post-hoc method KFLLLA) for 50 epochs with batch size 32 and with a initial learning rate 0.0005 which decays linearly to 0.00001 and we use the model from the final epoch for evaluation.

Image classification with CIFAR10 and CIFAR100 (Krizhevsky et al., 2009). For both CIFAR10 and CIFAR100 datasets, we randomly split the original training set into 4,5000 training and 5,000 validation instances, and test on the original 10,000 test instances. The input images are tokenized with a patch size of 4×4. For CIFAR10 without data augmentation, we use a ViT (Dosovitskiy et al., 2021) with 5 MHSA layers, each with 4 attention heads and a hidden dimension of 128. For all the other experiments, we use a ViT with 6 layers 4 attention heads, a hidden dimension of 256. We train all models (except post-hoc methods, TS and KFLLLA) except SGPA for 600 epochs with batch size 100 and with initial learning rate of 0.0005 which decays linearly to 0.00001. For SGPA, we use 32 global inducing points for each head, and we use the parameters from the 100th epoch of MLE to initialize the deep kernel hyperparameters, and continue training for 500 epochs. The best model is selected based on the validation accuracy computed every 10 epochs. For experiments with data augmentation, we consider the same data augmentation strategy (3-Augment) as in Touvron et al. (2022).

Graph property regression with ZINC (Dwivedi et al., 2020). The results are presented in B.3.2 which are averaged from 3 independent runs. We use the same split as in (Dwivedi et al., 2020), resulting in 10,000 training, 1,000 validation, and 1,000 test instances. Instead of applying graph-specific modifications to the network architecture, we use the feature engineering technique proposed in Kim et al.

(2022) to transform each graph into a sequence of input embeddings. We consider Transformers with 8 layer and 8 attention heads, same embedding dimension and hidden dimension of 80. For SGPA we use 10 global inducing points for each head. We train all models for 500 epochs with batch size 64 and with an initial learning rate 0.0004 which decays linearly to 0.000002. The best model is selected based on the validation accuracy computed at the end of every 10 epochs.

B.5 Running Time

We analyze the wall-clock training and inference time of SGPA here. In Table B.2, we present the computational time for a single batch at the inference stage with 10 Monte Carlo samples for CoLA (batch size = 227) and CIFAR10 (batch size = 200) (results obtained using a single Nvidia GTX 2080 Ti GPU card). For SGPA, we first pay an one-off cost of inverting the kernel matrices related to global inducing points ($\boldsymbol{K}_{\boldsymbol{k}_g\boldsymbol{k}_g}^{-1}$). Once this is done, we treat them as constant matrices and plug them in Eq. 3.15. When generating samples that are passed to the next layer, we diagonalize the covariance ($\boldsymbol{\Sigma}_d$) in Eq. 3.15 to avoid the costly computation of the Cholesky factor of $\boldsymbol{\Sigma}_d$.

The computational cost depends on the number of global inducing points used. For CoLA, we only used 5 global inducing points for each head, and the relative difference between inference times for MCD and SGPA is less than that for CIFAR10, where we use 32 global inducing points for each head. It is noteworthy that we haven't done extensive hyperparameter tuning for the number of global inducing points. It is likely that SGPA can still work well with a smaller number of global inducing points. For example, for CIFAR10, we later on also trained an SGPA model with 16 global inducing points for each head, and we did not see a considerable performance drop (Accuracy: 0.7790, NLL: 0.7259, ECE: 0.0625, MCE: 0.0819). In this case, the inference time can be further reduced from 0.986 to 0.807.

Table B.2: The computational time (in s) for a single batch at the inference stage with 10 Monte Carlo samples for CoLA (batch size = 227) and CIFAR10 (batch size = 200) (results obtained using a single Nvidia GTX 2080 Ti GPU card).

	MCD	MFVI	SGPA
CoLA	1.839	1.882	2.358
CIFAR10 (32 $\boldsymbol{k}_g$)	0.234	0.395	0.986
CIFAR10 (16 $\boldsymbol{k}_g$)	0.234	0.395	0.807

In Table B.3 we present the training time (in s) of one epoch for SGPA and MLE on CoLA (batch size = 32) and CIFAR10 (batch size = 100) (results obtained using a single Nvidia GTX 2080 Ti GPU card):

Table B.3: The training time (in s) of one epoch for SGPA and MLE on CoLA (batch size = 32) and CIFAR10 (batch size = 100) (results obtained using a single Nvidia GTX 2080 Ti GPU card).

	MLE	SGPA
CoLA	17.834	20.089
CIFAR10 (32 $\boldsymbol{k}_g$)	30.593	138.609
CIFAR10 (16 $\boldsymbol{k}_g$)	30.593	109.298

B.6 Connection with Sparse within Sparse GP

Unlike standard sparse GPs where the inducing points are shared across all inputs, SGPA consists of a set of input-dependent (or "amortized") inducing locations $\{\boldsymbol{k}_a^{l,h}\}$ and the corresponding variational parameters, which means we are using a different mean function for each input sequence. Consequently, SGPA based Transformers can not perfectly model the correlation between input sequences. Instead, they can only provide marginal uncertainty for each input sequence. Nevertheless, empirically we found that correlation might not be critical in applications such as text or image classification.

Interestingly, SGPA can be considered as an instantiation of sparse-within-sparse Gaussian process (SWSGP) (Tran et al., 2021; Jafrasteh et al., 2022), which allows adaptive inducing points for each input. Suppose the index set (in our case the embedding space) is $\boldsymbol{\chi}$, and $p(\boldsymbol{Z})$ is a distribution over M-element subset of $\boldsymbol{\chi}$ (ie. each random draw will give us M inducing locations from $\boldsymbol{\chi}$). The joint prior and approximate posterior become $p(\boldsymbol{f}, \boldsymbol{u}, \boldsymbol{Z}) = p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u}|\boldsymbol{Z})p(\boldsymbol{Z})$, and $q(\boldsymbol{u}, \boldsymbol{Z}) = q(\boldsymbol{u}|\boldsymbol{Z})p(\boldsymbol{Z})$, respectively. In Tran et al. (2021), for each input $\boldsymbol{x}$, they propose to select its M-nearest neighbors taken from the training inputs as inducing locations, so that $q(\boldsymbol{Z})$ is a delta distribution conditioned on $\boldsymbol{x}$, and $q(\boldsymbol{u}|\boldsymbol{Z})$ is the marginal variational distribution over function values evaluated at the selected inducing locations. In contrast, for each input sequence $\boldsymbol{x}$, in layer l, the inducing locations used by Transformer based on SGPA consist both input-dependent ones ($\{\boldsymbol{k}_a^{l,h}\}_{h=1}^H$), which are obtained from $\boldsymbol{x}$ using neural network as in Jafrasteh et al. (2022), and global inducing locations $\{\boldsymbol{k}_g^{l,h}\}_{h=1}^H$, which are shared across all input sequences.

Note that the input-dependent inducing points used during test time may not be encountered in training. Instead, we rely on the learned neural network to amortize them (Jafrasteh et al., 2022). Therefore the fitted mean function may not be consistent when test sequence includes tokens far away from the tokens in training sequences (as $\boldsymbol{v}_a$ may not be consistent). Empirically, we found this inconsistency issue to be minor for in-distribution test sequences. Furthermore, we argue that for OOD inputs, although the fitted posterior mean might be unreliable, the uncertainty

still increases since the posterior covariance is fully determined by the global inducing points which exhibits no inconsistency issue. Intuitively, the global keys $\{\boldsymbol{k}_g^{l,h}\}_{l=1,h=1}^{L,H}$, shared across all input sequences, play a similar role as the inducing locations in standard SVGP: they summarize the training set but focus on the uncertainty behavior only. As a result, the posterior variance in Eq. 3.15 still increases for queries that are less similar to the global keys as measured by the kernel. By propagating the uncertainty through each layer, we can still obtain increased uncertainty for input sequences that are very different from the training data, so that users can still be notified "when the model does not know" (Gal, 2016).

B.7 Results in Tables

We present numerical results (mean±standard error) for all experiments in tables.

We show in-distribution results in Tables B.4 to B.10.

We show OOD robustness results in Tables B.11 to B.27.

We show OOD detection results in Tables B.28 to B.32.

Table B.4: In-distribution performance: sentiment analysis with IMDB

	Model	Accuracy	NLL	ECE	MCE
SDP	MLE	0.8806±0.0016	0.3267±0.0068	0.0548±0.0029	0.1432±0.0093
Kernel	MLE	0.8822±0.0019	0.3202±0.0136	0.0519±0.0055	0.1568±0.0273
	TS	0.8822±0.0019	0.3202±0.0136	0.0519±0.0055	0.1568±0.0004
	MFVI	0.8799±0.0022	0.3476±0.0230	0.0585±0.0106	0.1493±0.0271
	MCD	0.8817±0.0014	0.2986±0.0070	0.0285±0.0058	0.0910±0.0218
	KFLLLA	0.8822±0.0019	0.4366±0.0015	0.1905±0.0034	0.2263±0.0048
	SNGP	0.8783±0.0013	0.3499±0.0080	0.0646±0.0037	0.1741±0.0106
	SGPA	0.8875±0.0004	0.2823±0.0027	0.0215±0.0034	0.0647±0.0105

Table B.5: In-distribution performance: linguistic acceptability with CoLA

	Model	MCC	NLL	ECE	MCE
SDP	MLE	25.0408±0.8223	1.8976±0.0223	0.2654±0.0039	0.3887±0.0230
Kernel	MLE	26.0722±0.4300	2.0083±0.0432	0.2643±0.0077	0.3702±0.0217
	MFVI	21.0864±1.8124	1.0809±0.0092	0.2018±0.0080	0.266±0.0113
	MCD	26.4399±0.4939	0.9121±0.0154	0.2113±0.0049	0.3233±0.0066
	KFLLLA	26.0295±0.4050	0.6023±0.0077	0.0241±0.0044	0.1063±0.0204
	SNGP	27.8715±1.2584	1.2075±0.0418	0.2325±0.0067	0.3230±0.0146
	SGPA	27.2380±1.1188	0.9070±0.0207	0.2076±0.0082	0.3062±0.0097

Table B.6: In-distribution performance: image classification for CIFAR10 without data augmentation

	Model	Accuracy	NLL	ECE	MCE
SDP	MLE	0.7439±0.0015	2.2940±0.0277	0.2703±0.0021	0.4509±0.0079
Kernel	MLE	0.7811±0.0019	1.3395±0.0135	0.2082±0.0010	0.3724±0.0103
	TS	0.7811±0.0019	0.8254±0.0077	0.1357±0.0012	0.2499±0.0081
	MFVI	0.7202±0.0024	0.7995±0.0073	0.0331±0.0023	0.0664±0.0064
	MCD	0.7910±0.0015	0.7789±0.0063	0.0799±0.0012	0.1501±0.0055
	KFLLLA	0.7752±0.0019	0.7201±0.0242	0.0710±0.0067	0.1497±0.0217
	SNGP	0.7837±0.0025	0.7534±0.0031	0.0745±0.0003	0.1443±0.0049
	SGPA	0.7787±0.0024	0.6968±0.0032	0.0589±0.0012	0.0825±0.0066
	DE	0.8235	0.6934	0.0504	0.0699
	SGPAE	0.8172	0.5657	0.0184	0.0579

Table B.7: In-distribution performance: image classification for CIFAR10 with data augmentation

	Model	Accuracy	NLL	ECE	MCE
SDP	MLE	0.8898±0.0009	0.6073±0.0330	0.1526±0.0043	0.3250±0.0118
Kernel	MLE	0.8442±0.0057	0.4700±0.0127	0.0408±0.0031	0.0951±0.0116
	TS	0.8442±0.0057	0.4552±0.0176	0.0326±0.0028	0.0532±0.0077
	MFVI	0.6967±0.0022	0.8643±0.0078	0.0428±0.0072	0.0635±0.0083
	MCD	0.8437±0.0074	0.4541±0.0190	0.0122±0.0013	0.0428±0.0049
	KFLLLA	0.8445±0.0057	0.4554±0.0158	0.0279±0.0022	0.0570±0.0090
	SNGP	0.8286±0.0118	0.4912±0.0351	0.0076±0.0007	0.0398±0.0051
	SGPA	0.8475±0.0048	0.4489±0.0121	0.0086±0.0013	0.0395±0.0037
	DE	0.8654	0.4062	0.0079	0.0724
	SGPAE	0.8691	0.3885	0.0068	0.0421

Table B.8: In-distribution performance: image classification for CIFAR100 without data augmentation

	Model	Accuracy	NLL	ECE	MCE
SDP	MLE	0.4893±0.0022	5.7677±0.0492	0.5276±0.0021	0.6216±0.0076
Kernel	MLE	0.5216±0.0100	4.3898±0.0829	0.4061±0.0023	0.5696±0.0093
	TS	0.5216±0.0100	2.5863±0.0527	0.2769±0.0013	0.4067±0.0032
	MFVI	0.4117±0.0015	2.4195±0.0113	0.0753±0.0002	0.1021±0.0057
	MCD	0.5341±0.0096	2.7122±0.0329	0.1791±0.0013	0.2651±0.0116
	KFLLLA	0.5084±0.0103	3.2984±0.0210	0.1682±0.0054	0.7690±0.0159
	SNGP	0.4715±0.0039	2.8296±0.0338	0.2451±0.0005	0.3028±0.0082
	SGPA	0.5302±0.0071	2.5643±0.0813	0.1730±0.0013	0.2463±0.0071
	DE	0.5815	2.4887	0.1026	0.1381
	SGPAE	0.5700	1.9434	0.0467	0.0704

Table B.9: In-distribution performance: image classification for CIFAR100 with data augmentation

	Model	Accuracy	NLL	ECE	MCE
SDP	MLE	0.5984±0.0024	1.8743±0.2032	0.1879±0.0006	0.2792±0.0512
Kernel	MLE	0.6053±0.0048	1.5254±0.0195	0.1066±0.0036	0.1578±0.0159
	TS	0.6053±0.0048	1.4512±0.0251	0.0303±0.0027	0.0825±0.0109
	MFVI	0.4496±0.0045	2.1132±0.0105	0.0527±0.0063	0.1339±0.0156
	MCD	0.6149±0.0074	1.4255±0.0275	0.0231±0.0032	0.0468±0.0056
	KFLLLA	0.5994±0.0038	1.4661±0.0206	0.0198±0.0022	0.0474±0.0071
	SNGP	0.5645±0.0106	1.5839±0.0416	0.0254±0.0018	0.0424±0.0060
	SGPA	0.6154±0.0010	1.4486±0.0055	0.0364±0.0055	0.0612±0.0067
	DE	0.6471	1.3023	0.0565	0.0986
	SGPEA	0.65	1.2761	0.0191	0.0593

Table B.10: In-distribution performance: graph property regression with ZINC dataset

	Model	MAE	RMSE	NLL
SDP	MLE	0.5733±0.0084	1.1459±0.0139	1.1459±0.0139
Kernel	MLE	0.5191±0.0344	1.0977±0.0869	1.0977±0.0869
	MCD	0.4385±0.0163	0.8369±0.0308	0.9264±0.0317
	MFVI	0.5702±0.0086	1.1050±0.0115	1.1683±0.0101
	SGPA	0.4341±0.0142	0.8199±0.0195	0.9319±0.0242
	DE	0.4711	0.8635	1.0949
	SGPAE	0.4557	0.8479	1.0118

Table B.11: OOD robustness: linguistic acceptability with CoLA dataset

	Model	MCC	NLL	ECE	MCE
SDP	MLE	19.4835±2.0234	2.0574±0.0516	0.2914±0.0098	0.4913±0.0418
Kernel	MLE	18.6467±1.6116	2.3565±0.0974	0.2909±0.0067	0.3864±0.0303
	MFVI	16.6363±1.5793	1.1349±0.0445	0.2184±0.0035	0.3289±0.0151
	MCD	20.4774±1.4727	1.0114±0.0234	0.2370±0.0075	0.3439±0.0177
	KFLLLA	18.6250±1.6321	0.6299±0.0067	0.0361±0.0065	0.0876±0.0065
	SNGP	23.6410±1.5986	1.3070±0.0412	0.2537±0.0049	0.3828±0.0189
	SGPA	22.9190±1.5986	0.9514±0.0261	0.2257±0.0069	0.3399±0.0294

Table B.12: Accuracy of CIFAR10-C for ViTs trained on clean data without data augmentation.

		Skew Intensity				
	Model	1	2	3	4	5
SDP	MLE	0.7120±0.0011	0.6704±0.0011	0.6359±0.0014	0.5883±0.0017	0.5354±0.0015
Kernel	MLE	0.7254±0.0019	0.6632±0.0028	0.6139±0.0023	0.5494±0.0028	0.4863±0.0027
	MFVI	0.6606±0.0093	0.6109±0.0083	0.5730±0.0077	0.5266±0.0073	0.4778±0.0065
	MCD	0.7327±0.0013	0.6717±0.0015	0.6229±0.0016	0.5603±0.0019	0.4982±0.0020
	KFLLLA	0.7186±0.0014	0.6580±0.0035	0.6092±0.0038	0.5462±0.0042	0.4825±0.0038
	SNGP	0.7281±0.0024	0.6676±0.0025	0.6208±0.0020	0.5622±0.0020	0.5006±0.0019
	SGPA	0.7175±0.0038	0.6559±0.0042	0.6067±0.0040	0.5466±0.0033	0.4832±0.0046
	DE	0.7700	0.7089	0.6585	0.5908	0.5246
	SGPAE	0.7516	0.6911	0.6401	0.5779	0.5173

Table B.13: Accuracy of CIFAR10-C for ViTs trained on clean data with data augmentation.

		Skew Intensity				
	Model	1	2	3	4	5
SDP	MLE	0.8516±0.0010	0.8139±0.0011	0.7752±0.0010	0.7165±0.0013	0.6500±0.0012
Kernel	MLE	0.7944±0.0061	0.7499±0.0074	0.7090±0.0082	0.6465±0.0085	0.5807±0.0094
	MFVI	0.6605±0.0016	0.6222±0.0024	0.5875±0.0028	0.5377±0.0032	0.4913±0.0040
	MCD	0.7939±0.0068	0.7480±0.0082	0.7052±0.0089	0.6415±0.0092	0.5751±0.0097
	KFLLLA	0.7949±0.0030	0.7508±0.0037	0.7100±0.0041	0.6480±0.0043	0.5821±0.0047
	SNGP	0.7783±0.0062	0.7336±0.0069	0.6932±0.0069	0.6314±0.0062	0.5710±0.0057
	SGPA	0.7910±0.0027	0.7426±0.0021	0.6946±0.0075	0.6356±0.0033	0.5712±0.0020
	DE	0.8237	0.7819	0.7428	0.6795	0.6130
	SGPAE	0.8105	0.7646	0.7220	0.6581	0.5914

Table B.14: Accuracy of CIFAR100-C for ViTs trained on clean data without data augmentation.

		Skew Intensity				
	Model	1	2	3	4	5
SDP	MLE	0.3947±0.0022	0.3484±0.0018	0.3223±0.0014	0.2805±0.0013	0.2402±0.0009
Kernel	MLE	0.4456±0.0075	0.3769±0.0058	0.3373±0.0049	0.2835±0.0034	0.2339±0.0029
	MFVI	0.3380±0.0020	0.2832±0.0022	0.2540±0.0018	0.2155±0.0014	0.1803±0.0015
	MCD	0.4526±0.0074	0.3840±0.0061	0.3446±0.0052	0.2913±0.0037	0.2403±0.0031
	KFLLLA	0.4350±0.0070	0.3656±0.0053	0.3251±0.0045	0.2717±0.0030	0.2229±0.0025
	SNGP	0.4053±0.0020	0.3455±0.0014	0.3093±0.0015	0.2624±0.0015	0.2203±0.0011
	SGPA	0.4472±0.0050	0.3764±0.0035	0.3371±0.0027	0.2828±0.0021	0.2317±0.0022
	DE	0.5054	0.4289	0.3858	0.3234	0.2690
	SGPAE	0.4848	0.4104	0.3675	0.3087	0.2539

Table B.15: Accuracy of CIFAR100-C for ViTs trained on clean data with data augmentation.

		Skew Intensity				
	Model	1	2	3	4	5
SDP	MLE	0.5384±0.0030	0.4854±0.0038	0.4464±0.0041	0.3930±0.0037	0.3410±0.0029
Kernel	MLE	0.5327±0.0018	0.4700±0.0018	0.4264±0.0017	0.3690±0.0017	0.3134±0.0013
	MFVI	0.3991±0.0019	0.3535±0.0017	0.3224±0.0019	0.2836±0.0017	0.2444±0.0014
	MCD	0.5397±0.0020	0.4775±0.0018	0.4349±0.0015	0.3777±0.0011	0.3213±0.0010
	KFLLLA	0.5196±0.0019	0.4577±0.0018	0.4148±0.0017	0.3581±0.0014	0.3035±0.0014
	SNGP	0.4979±0.0038	0.4440±0.0035	0.4042±0.0034	0.3534±0.0029	0.3066±0.0023
	SGPA	0.5189±0.0054	0.4603±0.0010	0.4181±0.0013	0.3625±0.0014	0.3061±0.0014
	DE	0.5742	0.5104	0.4649	0.4039	0.3449
	SGPAE	0.5592	0.4936	0.4476	0.3908	0.3316

Table B.16: NLL of CIFAR10-C for ViTs trained on clean data without data augmentation.

		Skew Intensity				
	Model	1	2	3	4	5
SDP	MLE	2.6411±0.0202	3.1620±0.0202	3.6449±0.0231	4.3843±0.0283	5.2626±0.0287
Kernel	MLE	1.7906±0.0179	2.3460±0.0321	2.8899±0.0385	3.6899±0.0544	4.5003±0.0811
	MFVI	0.9761±0.0274	1.1345±0.0267	1.2737±0.0259	1.4705±0.0237	1.7038±0.0222
	MCD	1.1466±0.0099	1.5153±0.0168	1.8926±0.0234	2.4692±0.0315	3.1086±0.0431
	KFLLLA	0.9573±0.0498	1.2092±0.0577	1.4435±0.0683	1.7889±0.0937	2.1537±0.1198
	SNGP	1.0579±0.0066	1.3681±0.0115	1.6570±0.0129	2.0807±0.0136	2.5271±0.0146
	SGPA	0.9603±0.0128	1.2340±0.0213	1.4972±0.0283	1.8951±0.0383	2.3812±0.0586
	DE	0.9317	1.2424	1.5716	2.0911	2.6373
	SGPAE	0.7816	0.9985	1.2125	1.5365	1.9349

Table B.17: NLL of CIFAR10-C for ViTs trained on clean data with data augmentation.

		Skew Intensity				
	Model	1	2	3	4	5
SDP	MLE	0.8306±0.0422	1.0712±0.0559	1.3665±0.0753	1.8795±0.1085	2.5810±0.1571
Kernel	MLE	0.6364±0.0140	0.7960±0.0207	0.9614±0.0296	1.2404±0.0416	1.5818±0.0538
	MFVI	0.9616±0.0044	1.0706±0.0061	1.1802±0.0098	1.3612±0.0160	1.5629±0.0248
	MCD	0.6067±0.0180	0.7579±0.0236	0.9144±0.0304	1.1707±0.0363	1.4977±0.0420
	KFLLLA	0.6104±0.0080	0.7565±0.0108	0.9044±0.0144	1.1496±0.0183	1.4487±0.0230
	SNGP	0.6439±0.0187	0.7874±0.0218	0.9304±0.0225	1.1773±0.0223	1.4671±0.0225
	SGPA	0.6204±0.0054	0.7739±0.0054	0.9298±0.0047	1.1873±0.0118	1.5058±0.0226
	DE	0.5641	0.7028	0.8441	1.0737	1.3595
	SGPAE	0.5230	0.6503	0.7814	1.0027	1.2835

Table B.18: NLL of CIFAR100-C for ViTs trained on clean data without data augmentation.

		Skew Intensity				
	Model	1	2	3	4	5
SDP	MLE	7.4649±0.0475	8.5233±0.0524	9.1964±0.0527	10.4061±0.0555	11.8540±0.0649
Kernel	MLE	5.5126±0.0578	6.7077±0.0416	7.5503±0.0237	9.0093±0.0353	10.5647±0.0699
	MFVI	3.0138±0.0132	3.4762±0.0136	3.7921±0.0135	4.3295±0.0153	4.9042±0.0200
	MCD	3.8996±0.0266	4.8412±0.0178	5.5397±0.0269	6.7984±0.0591	8.2454±0.0959
	KFLLLA	3.4366±0.0172	3.5660±0.0160	3.6457±0.0152	3.7645±0.0135	3.8862±0.0115
	SNGP	3.6404±0.0297	4.3067±0.0360	4.7994±0.0446	5.6242±0.0502	6.5029±0.0504
	SGPA	3.6565±0.1290	4.5188±0.1401	5.1522±0.1510	6.3096±0.1784	7.6306±0.2161
	DE	3.1930	4.0004	4.6072	5.7357	6.9765
	SGPAE	2.8407	3.5393	4.0466	4.9890	6.0803

Table B.19: NLL of CIFAR100-C for ViTs trained on clean data with data augmentation.

		Skew Intensity				
	Model	1	2	3	4	5
SDP	MLE	2.2595±0.2276	2.6313±0.2735	2.9786±0.3148	3.5393±0.3720	4.1865±0.4426
Kernel	MLE	1.9695±0.0167	2.3721±0.0260	2.7423±0.0346	3.3355±0.0500	3.9813±0.0602
	MFVI	2.3501±0.0073	2.5702±0.0084	2.7571±0.0112	3.0582±0.0123	3.3805±0.0147
	MCD	1.8337±0.0048	2.1874±0.0073	2.5076±0.0112	3.0263±0.0205	3.6196±0.0273
	KFLLLA	1.8645±0.0092	2.2008±0.0095	2.4935±0.0095	2.9465±0.0112	3.4418±0.0135
	SNGP	1.9167±0.0161	2.2067±0.0162	2.4753±0.0163	2.9166±0.0133	3.3598±0.0116
	SGPA	1.9307±0.0075	2.3154±0.0104	2.6598±0.0130	3.2105±0.0151	3.8575±0.0180
	DE	1.7354	2.0787	2.3857	2.8802	3.4604
	SGPAE	1.6583	1.9839	2.2802	2.7592	3.2976

Table B.20: ECE of CIFAR10-C for ViTs trained on clean data without data augmentation.

		Skew Intensity				
	Model	1	2	3	4	5
SDP	MLE	0.2919±0.0021	0.3297±0.0021	0.3623±0.0023	0.3964±0.0026	0.4458±0.0030
Kernel	MLE	0.2514±0.0019	0.3003±0.0020	0.3396±0.0026	0.3898±0.0028	0.4355±0.0032
	MFVI	0.0324±0.0013	0.0767±0.0019	0.1031±0.0021	0.1273±0.0025	0.1684±0.0025
	MCD	0.1191±0.0011	0.1589±0.0013	0.1928±0.0013	0.2390±0.0015	0.2831±0.0017
	KFLLLA	0.1044±0.0012	0.1437±0.0017	0.1827±0.0017	0.2306±0.0029	0.2674±0.0035
	SNGP	0.1095±0.0011	0.1477±0.0013	0.1860±0.0015	0.2337±0.0015	0.2724±0.0018
	SGPA	0.0894±0.0013	0.1305±0.0018	0.1653±0.0020	0.2137±0.0024	0.2562±0.0024
	DE	0.0771	0.1095	0.1406	0.1826	0.2168
	SGPAE	0.0323	0.0644	0.0946	0.1381	0.1736

Table B.21: ECE of CIFAR10-C for ViTs trained on clean data with data augmentation.

		Skew Intensity				
	Model	1	2	3	4	5
SDP	MLE	0.1731±0.0015	0.1962±0.0015	0.2094±0.0017	0.2582±0.0021	0.3046±0.0031
Kernel	MLE	0.0666±0.0014	0.0920±0.0016	0.1170±0.0018	0.1605±0.0027	0.2090±0.0036
	MFVI	0.0295±0.0006	0.0519±0.0008	0.0715±0.0015	0.1097±0.0019	0.1534±0.0026
	MCD	0.0245±0.0006	0.0433±0.0008	0.0657±0.0013	0.1084±0.0017	0.1545±0.0023
	KFAC-LLLA	0.0352±0.0006	0.0587±0.0008	0.0797±0.0014	0.1175±0.0018	0.1634±0.0024
	SNGP	0.0260±0.0004	0.0368±0.0009	0.0555±0.0016	0.0929±0.0025	0.1370±0.0035
	SGPA	0.0217±0.0004	0.0368±0.0009	0.0565±0.0017	0.0971±0.0027	0.1421±0.0038
	DE	0.0291	0.0369	0.0549	0.0904	0.1338
	SGPAE	0.0224	0.0351	0.0498	0.0821	0.1196

Table B.22: ECE of CIFAR100-C for ViTs trained on clean data without data augmentation.

		Skew Intensity				
	Model	1	2	3	4	5
SDP	MLE	0.5735±0.0032	0.6016±0.0028	0.6272±0.0031	0.6450±0.0027	0.6799±0.0024
Kernel	MLE	0.4588±0.0028	0.5095±0.0030	0.5393±0.0028	0.5826±0.0019	0.6209±0.0020
	MFVI	0.1014±0.0012	0.1458±0.0011	0.1592±0.0012	0.2092±0.0013	0.2575±0.0023
	MCD	0.2376±0.0010	0.2799±0.0009	0.3086±0.0017	0.3562±0.0023	0.4012±0.0031
	KFAC-LLLA	0.1753±0.0029	0.1740±0.0031	0.1553±0.0029	0.1478±0.0020	0.1427±0.0020
	SNGP	0.2427±0.0010	0.2839±0.0010	0.3236±0.0017	0.3621±0.0023	0.4056±0.0031
	SGPA	0.2337±0.0012	0.2763±0.0011	0.3045±0.0012	0.3520±0.0012	0.3963±0.0023
	DE	0.1696	0.2117	0.2400	0.2877	0.3283
	SGPAE	0.0972	0.1390	0.1665	0.2149	0.2605

Table B.23: ECE of CIFAR100-C for ViTs trained on clean data with data augmentation.

		Skew Intensity				
	Model	1	2	3	4	5
SDP	MLE	0.2179±0.0022	0.2485±0.0020	0.2901±0.0023	0.3327±0.0025	0.3816±0.0025
Kernel	MLE	0.1459±0.0020	0.1793±0.0021	0.2067±0.0022	0.2454±0.0023	0.2797±0.0023
	MFVI	0.0478±0.0032	0.0409±0.0038	0.0510±0.0040	0.0681±0.0042	0.0824±0.0044
	MCD	0.0614±0.0021	0.0919±0.0025	0.1197±0.0028	0.1596±0.0028	0.1979±0.0032
	KFAC-LLLA	0.0493±0.0045	0.0635±0.0057	0.0862±0.0053	0.1323±0.0045	0.1764±0.0047
	SNGP	0.0368±0.0019	0.0543±0.0025	0.0779±0.0027	0.1192±0.0025	0.1429±0.0031
	SGPA	0.0742±0.0040	0.1060±0.0045	0.1341±0.0046	0.1733±0.0044	0.2124±0.0045
	DE	0.0355	0.0562	0.0780	0.1144	0.1440
	SGPAE	0.0434	0.0511	0.0680	0.1002	0.1308

Table B.24: MCE of CIFAR10-C for ViTs trained on clean data without data augmentation.

		Skew Intensity				
	Model	1	2	3	4	5
SDP	MLE	0.4666±0.0027	0.4840±0.0026	0.5041±0.0018	0.5280±0.0019	0.5534±0.0020
Kernel	MLE	0.4038±0.0028	0.4362±0.0020	0.4655±0.0020	0.5051±0.0033	0.5442±0.0036
	MFVI	0.0598±0.0112	0.0852±0.0150	0.1125±0.0147	0.1586±0.0147	0.2011±0.0149
	MCD	0.1902±0.0021	0.2402±0.0025	0.2807±0.0026	0.3413±0.0036	0.3949±0.0041
	KFLLLA	0.1987±0.0251	0.2362±0.0241	0.2729±0.0238	0.3248±0.0220	0.3733±0.0210
	SNGP	0.1854±0.0018	0.2337±0.0021	0.2710±0.0011	0.3229±0.0021	0.3798±0.0008
	SGPA	0.1273±0.0042	0.1801±0.0046	0.2212±0.0061	0.2807±0.0057	0.3340±0.0047
	DE	0.1051	0.1402	0.1813	0.2476	0.3091
	SGPAE	0.0570	0.0816	0.1173	0.1757	0.2208

Table B.25: MCE of CIFAR10-C for ViTs trained on clean data with data augmentation.

		Skew Intensity				
	Model	1	2	3	4	5
SDP	MLE	0.3500±0.0125	0.3649±0.0126	0.3870±0.0123	0.4186±0.0122	0.4561±0.0129
Kernel	MLE	0.1188±0.0093	0.1451±0.0091	0.1776±0.0090	0.2341±0.0092	0.2886±0.0090
	MFVI	0.0772±0.0048	0.0965±0.0025	0.1283±0.0052	0.1903±0.0067	0.2261±0.0090
	MCD	0.0566±0.0032	0.0788±0.0031	0.1110±0.0048	0.1720±0.0062	0.2298±0.0057
	KFLLLA	0.0808±0.0031	0.1035±0.0028	0.1328±0.0030	0.1845±0.0036	0.2396±0.0036
	SNGP	0.0557±0.0013	0.0797±0.0029	0.1093±0.0039	0.1764±0.0040	0.2318±0.0044
	SGPA	0.0569±0.0022	0.0781±0.0026	0.1059±0.0030	0.1601±0.0054	0.2167±0.0050
	DE	0.0868	0.0855	0.1003	0.1334	0.1705
	SGPAE	0.0627	0.0659	0.0882	0.1262	0.1776

Table B.26: MCE of CIFAR100-C for ViTs trained on clean data without data augmentation.

		Skew Intensity				
	Model	1	2	3	4	5
SDP	MLE	0.6608±0.0014	0.6852±0.0014	0.7009±0.0022	0.7251±0.0020	0.7542±0.0011
Kernel	MLE	0.6037±0.0032	0.6428±0.0031	0.6649±0.0027	0.7037±0.0007	0.7385±0.0010
	MFVI	0.1925±0.0031	0.2594±0.0039	0.3069±0.0055	0.3809±0.0044	0.4562±0.0027
	MCD	0.3424±0.0042	0.4103±0.0047	0.4554±0.0045	0.5255±0.0052	0.5902±0.0038
	KFLLLA	0.7691±0.0165	0.7632±0.0181	0.7512±0.0194	0.7248±0.0146	0.6699±0.0228
	SNGP	0.3631±0.0042	0.4201±0.0044	0.4624±0.0039	0.5177±0.0043	0.5749±0.0037
	SGPA	0.3217±0.0057	0.3886±0.0083	0.4353±0.0093	0.5047±0.0083	0.5726±0.0072
	DE	0.1767	0.2257	0.2655	0.3522	0.4202
	SGPAE	0.1247	0.1766	0.2209	0.3142	0.3965

Table B.27: MCE of CIFAR100-C for ViTs trained on clean data with data augmentation.

		Skew Intensity				
	Model	1	2	3	4	5
SDP	MLE	0.3142±0.0470	0.3487±0.0458	0.3816±0.0451	0.4308±0.0412	0.4839±0.0387
Kernel	MLE	0.2237±0.0136	0.2650±0.0122	0.3098±0.0131	0.3733±0.0117	0.4381±0.0140
	MFVI	0.1330±0.0083	0.1173±0.0064	0.1275±0.0051	0.1751±0.0056	0.2208±0.0069
	MCD	0.0995±0.0066	0.1418±0.0067	0.1845±0.0067	0.2568±0.0081	0.3240±0.0082
	KFLLLA	0.0893±0.0040	0.1278±0.0048	0.1689±0.0057	0.2355±0.0067	0.2999±0.0076
	SNGP	0.0636±0.0012	0.0972±0.0013	0.1377±0.0015	0.2007±0.0017	0.2620±0.0012
	SGPA	0.1120±0.0022	0.1582±0.0034	0.2062±0.0040	0.2776±0.0028	0.3455±0.0030
	DE	0.0920	0.0971	0.1220	0.1906	0.2444
	SGPAE	0.0722	0.0888	0.1088	0.1766	0.2291

Table B.28: AUROC and AUPR metrics for OOD detection using ViTs trained on CIFAR10 without data augmentation.

		OOD: CIFAR100		OOD: SVHN		OOD: MINIIMAGENET	
	Model	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR
SDP	MLE	0.6893±0.0018	0.6433±0.0023	0.6983±0.0109	0.8206±0.0074	0.7115±0.0018	0.9219±0.0006
Kernel	MLE	0.7208±0.0029	0.6774±0.0029	0.7477±0.0118	0.8518±0.0068	0.7325±0.0016	0.9300±0.0004
	MFVI	0.7177±0.0055	0.6722±0.0050	0.7059±0.0195	0.7944±0.0134	0.7531±0.0036	0.9407±0.0008
	MCD	0.7426±0.0016	0.7046±0.0016	0.6860±0.0138	0.8045±0.0083	0.7443±0.0019	0.9356±0.0006
	KFLLLA	0.7228±0.0056	0.6889±0.0055	0.7599±0.0105	0.8345±0.0069	0.7855±0.0046	0.9487±0.0015
	SNGP	0.7391±0.0016	0.7023±0.0029	0.7135±0.0125	0.8258±0.0083	0.7391±0.0050	0.9341±0.0015
	SGPA	0.7498±0.0010	0.7111±0.0023	0.7198±0.0085	0.8125±0.0067	0.7501±0.0013	0.9382±0.0004
	DE	0.7757±0.0000	0.7423±0.0000	0.7819±0.0000	0.8512±0.0000	0.8007±0.0000	0.9520±0.0000
	SGPAE	0.7872±0.0000	0.7482±0.0000	0.7600±0.0000	0.8619±0.0000	0.8031±0.0000	0.9529±0.0000

Table B.29: AUROC and AUPR metrics for OOD detection using ViTs trained on CIFAR10 with data augmentation.

		OOD: CIFAR100		OOD: SVHN		OOD: MINIIMAGENET	
	Model	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR
SDP	MLE	0.8245±0.0007	0.7780±0.0013	0.8738±0.0020	0.9308±0.0015	0.8375±0.0009	0.9585±0.0005
Kernel	MLE	0.7908±0.0034	0.7523±0.0028	0.8748±0.0023	0.9338±0.0015	0.8148±0.0027	0.9555±0.0005
	MFVI	0.7009±0.0043	0.6530±0.0045	0.7652±0.0179	0.8468±0.0109	0.7506±0.0073	0.9379±0.0027
	MCD	0.7995±0.0038	0.7614±0.0030	0.8754±0.0040	0.9304±0.0029	0.8282±0.0029	0.9595±0.0007
	KFLLLA	0.7989±0.0041	0.7617±0.0036	0.8884±0.0036	0.9415±0.0022	0.8273±0.0034	0.9591±0.0007
	SNGP	0.7889±0.0092	0.7543±0.0104	0.8776±0.0044	0.9326±0.0032	0.8125±0.0072	0.9551±0.0020
	SGPA	0.8007±0.0014	0.7630±0.0013	0.8746±0.0061	0.9281±0.0043	0.8319±0.0014	0.9600±0.0004
	DE	0.8230±0.0000	0.7869±0.0000	0.9195±0.0000	0.9575±0.0000	0.8555±0.0000	0.9666±0.0000
	SGPAE	0.8264±0.0000	0.7917±0.0000	0.9004±0.0000	0.9452±0.0000	0.8606±0.0000	0.9677±0.0000

Table B.30: AUROC and AUPR metrics for OOD detection using ViTs trained on CIFAR100 without data augmentation.

		OOD: CIFAR10		OOD: SVHN		OOD: MINIIMAGENET	
	Model	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR
SDP	MLE	0.6091±0.0023	0.5756±0.0030	0.6242±0.0014	0.7856±0.0015	0.6512±0.0018	0.9042±0.0007
Kernel	MLE	0.6355±0.0052	0.5983±0.0045	0.6725±0.0111	0.8171±0.0062	0.6582±0.0035	0.9071±0.0010
	MFVI	0.6277±0.0036	0.5900±0.0037	0.6242±0.0108	0.7729±0.0079	0.6496±0.0054	0.9051±0.0024
	MCD	0.6729±0.0069	0.6310±0.0076	0.6054±0.0057	0.7597±0.0038	0.6859±0.0014	0.9155±0.0005
	KFLLLA	0.6297±0.0059	0.5920±0.0071	0.6733±0.0159	0.8101±0.0086	0.6915±0.0057	0.9194±0.0017
	SNGP	0.6448±0.0016	0.6080±0.0020	0.6764±0.0096	0.8093±0.0063	0.6823±0.0023	0.9161±0.0009
	SGPA	0.6712±0.0052	0.6297±0.0057	0.6454±0.0147	0.7854±0.0099	0.6911±0.0028	0.9170±0.0009
	DE	0.6848±0.0000	0.6377±0.0000	0.7388±0.0000	0.8425±0.0000	0.7394±0.0000	0.9300±0.0000
	SGPAE	0.6925±0.0000	0.6461±0.0000	0.6961±0.0000	0.8213±0.0000	0.7398±0.0000	0.9304±0.0000

Table B.31: AUROC and AUPR metrics for OOD detection using ViTs trained on CIFAR100 with data augmentation.

		OOD: CIFAR10		OOD: SVHN		OOD: MINIIMAGENET	
	Model	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR
SDP	MLE	0.6778±0.0022	0.6312±0.0001	0.7652±0.0093	0.8769±0.0055	0.7430±0.0065	0.9346±0.0020
Kernel	MLE	0.6899±0.0026	0.6486±0.0023	0.7570±0.0056	0.8670±0.0025	0.7536±0.0038	0.9374±0.0012
	MFVI	0.6403±0.0042	0.5924±0.0035	0.7485±0.0106	0.8640±0.0068	0.7269±0.0099	0.9317±0.0030
	MCD	0.7047±0.0025	0.6568±0.0029	0.7972±0.0040	0.8885±0.0023	0.7780±0.0023	0.9440±0.0008
	KFLLLA	0.6924±0.0011	0.6510±0.0022	0.8164±0.0043	0.9104±0.0053	0.7726±0.0084	0.9435±0.0024
	SNGP	0.6813±0.0034	0.6388±0.0034	0.7616±0.0090	0.8701±0.0052	0.7486±0.0048	0.9367±0.0014
	SGPA	0.7056±0.0012	0.6586±0.0011	0.8120±0.0035	0.8971±0.0027	0.7755±0.0012	0.9430±0.0004
	DE	0.7171±0.0000	0.6718±0.0000	0.8459±0.0000	0.9173±0.0000	0.8076±0.0000	0.9517±0.0000
	SGPAE	0.7198±0.0000	0.6719±0.0000	0.8686±0.0000	0.9298±0.0000	0.8152±0.0000	0.9536±0.0000

Table B.32: AUROC and AUPR metrics for OOD detection using Transformers trained on ZINC.

	Model	AUROC	AUPR
SDP	MCD	0.4566±0.0072	0.4889±0.0029
Kernel	MFVI	0.7254±0.0632	0.6821±0.0512
	MCD	0.8797±0.0739	0.8629±0.0571
	SGPA	0.8968±0.0317	0.8705±0.0335
	DE	0.9783±0.0000	0.9697±0.0000
	SGPAE	0.9884±0.0000	0.9727±0.0000

Appendix C

Supplementary Material for Chapter 4

C.1 Computing the Prior Inducing Covariance $\mathbf{K}_{\mathbf{uu}}^{(t)}$ as Direct ODE Evolution

Although in the experiments in the main text, we compute $\boldsymbol{K}_{\boldsymbol{uu}}^{(t)}$ based on RFF approximation, here for completeness, we provide the detailed derivation for an alternative approach of computing it, which directly evolves $\boldsymbol{K}_{\boldsymbol{uu}}^{(t)}$ as a matrix ODE, when the inducing functions are defined via HiPPO-LegS. Recall that

$$[\boldsymbol{K}_{\boldsymbol{uu}}^{(t)}]_{\ell,m} = \iint k(x,x')\phi_\ell^{(t)}(x)\phi_m^{(t)}(x')\mathrm{d}x\mathrm{d}x', \tag{C.1}$$

where $\phi_\ell^{(t)}(x) = g_\ell^{(t)}(x)\,\omega^{(t)}(x)$ are the time-varying basis functions under the HiPPO-LegS framework.

Differentiating $[\boldsymbol{K}_{\boldsymbol{uu}}^{(t)}]_{\ell,m}$ with respect to t gives

$$\frac{d}{dt}\,[\boldsymbol{K}_{\boldsymbol{uu}}^{(t)}]_{\ell,m} = \iint k(x,x')\frac{\partial}{\partial t}\left[\phi_\ell^{(t)}(x)\phi_m^{(t)}(x')\right]\mathrm{d}x\mathrm{d}x'. \tag{C.2}$$

Applying the product rule:

$$\frac{\mathrm{d}}{\mathrm{d}t}[\boldsymbol{K}_{\boldsymbol{uu}}^{(t)}]_{\ell,m} = \iint k(x,x')\frac{\partial}{\partial t}\phi_\ell^{(t)}(x)\phi_m^{(t)}(x')\mathrm{d}x\mathrm{d}x' + \iint k(x,x')\phi_\ell^{(t)}(x)\frac{\partial}{\partial t}\phi_m^{(t)}(x')\mathrm{d}x\mathrm{d}x'. \tag{C.3}$$

In HiPPO-LegS, each $\phi_\ell^{(t)}(x)$ obeys an ODE governed by lower order scaled Legendre polynomials on $[0,t]$ and a Dirac delta boundary term at $x = t$, as we show in

Section 2.1.2. Concretely,

$$\frac{\partial}{\partial t}\phi_\ell^{(t)}(x) = -\frac{\sqrt{2\ell+1}}{t}\Big[\frac{\ell+1}{\sqrt{2\ell+1}}\phi_\ell^{(t)}(x) + \sqrt{2\ell-1}\phi_{\ell-1}^{(t)}(x) + \sqrt{2\ell-3}\phi_{\ell-2}^{(t)}(x)\cdots\Big] + \frac{1}{t}\,\delta_t(x), \tag{C.4}$$

where $\delta_t(x)$ is the Dirac delta at $x = t$.

Substituting this expression into the integrals yields the boundary terms of the form $\int k(t,x')\,\phi_m^{(t)}(x')\,\mathrm{d}x'$, along with lower-order terms involving $\{[\boldsymbol{K}_{\boldsymbol{uu}}^{(t)}]_{\ell,m}, [\boldsymbol{K}_{\boldsymbol{uu}}^{(t)}]_{\ell-1,m}, \cdots\}$, etc. Summarizing in matrix form leads to

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{K}_{\boldsymbol{uu}}^{(t)} = \left[\boldsymbol{A}(t)\,\boldsymbol{K}_{\boldsymbol{uu}}^{(t)} + \boldsymbol{K}_{\boldsymbol{uu}}^{(t)}\boldsymbol{A}(t)^\top\right] + \frac{1}{t}\left[\tilde{\boldsymbol{B}}(t) + \tilde{\boldsymbol{B}}(t)^\top\right], \tag{C.5}$$

where the lm-th entry of $\boldsymbol{K}_{\boldsymbol{uu}}^{(t)} \in \mathbb{R}^{M\times M}$ is $[\boldsymbol{K}_{\boldsymbol{uu}}^{(t)}]_{\ell,m}$, $\boldsymbol{A}(t) \in \mathbb{R}^{M\times M}$ is the same lower-triangular matrix from the HiPPO-LegS framework defined in Eq. 2.14, and $\tilde{\boldsymbol{B}}(t) \in \mathbb{R}^{M\times M}$ is built from the boundary contributions as

$$\tilde{\boldsymbol{B}}(t) = \boldsymbol{c}(t)\mathbf{1}_M, \tag{C.6}$$

where $\mathbf{1}_M \in \mathbb{R}^{1\times M}$ is a row vector of ones of size M and $\boldsymbol{c}(t) \in \mathbb{R}^{M\times 1}$ is the coefficient vector with each element being

$$[\boldsymbol{c}(t)]_\ell = \int k\,(t,x)\;\phi_\ell^{(t)}(x)\,\mathrm{d}x. \tag{C.7}$$

After discretizing in t (e.g. an Euler scheme), one can recurrently update $\boldsymbol{K}_{\boldsymbol{uu}}^{(t)}$ and the boundary vector $\boldsymbol{c}(t)$ over time.

Unstability of directly evolving $\mathbf{K}_{\mathbf{uu}}^{(t)}$ as ODE. Empirically, we find that the direct ODE approach is less stable compared with RFF approach. Intuitively, it can be seen from the difference in the forms of their evolutions, especially in the first term. In RFF approach, the first term of the evolution of Fourier feature is of the form $\boldsymbol{A}(t)\,\boldsymbol{Z}_w^{(t)}$, which includes evolving vectors with the operator $\mathcal{L}_1 : \boldsymbol{X} \to \boldsymbol{A}(t)\boldsymbol{X}$. In direct ODE approach, the first term in the direct evolution of $\boldsymbol{K}_{\boldsymbol{uu}}^{(t)}$ is of the form $\boldsymbol{A}(t)\,\boldsymbol{K}_{\boldsymbol{uu}}^{(t)} + \boldsymbol{K}_{\boldsymbol{uu}}^{(t)}\boldsymbol{A}(t)^\top$, which requires the Lyapunov operator $\mathcal{L}_2 : \boldsymbol{X} \to \boldsymbol{A}(t)\boldsymbol{X} + \boldsymbol{X}\boldsymbol{A}(t)^\top$. The critical difference is that $\mathcal{L}_2$ has eigenvalues $\{\lambda_i + \lambda_j\}$ (where λ_i and λ_j are eigenvalues of $\boldsymbol{A}(t)$) (Horn and Johnson, 1991), while $\mathcal{L}_1$ has eigenvalues $\{\lambda_i\}$. Since HiPPO-LegS uses a lower-triangular $\boldsymbol{A}(t)$ with negative diagonal entries, the eigenvalues are all negative $\lambda_i < 0$. Hence, the eigenvalues of the Lyapunov operator $\mathcal{L}_2$ are approximately as twice negative as the eigenvalues of $\mathcal{L}_1$, leading to a stiff ODE system with poorer numerical conditioning.

C.2 Finite Basis Approximation of Posterior OHSVGP

Here, we show that $q(\boldsymbol{u}^{(t)})$ is the distribution of HiPPO coefficients of the posterior OHSVGP $q_t(f)$. From Eq. 2.36, the posterior of the function values evaluated at arbitrary indices $\boldsymbol{X}$ is $q_t(\boldsymbol{f_X}) = \mathcal{N}(\boldsymbol{f_X}; \mathbf{K}^{(t)}_{\boldsymbol{f_X}\mathbf{u}}\mathbf{K}^{(t)-1}_{\mathbf{uu}}\boldsymbol{m}^{(t)}_{\mathbf{u}}, \mathbf{K}_{\boldsymbol{f_X}\boldsymbol{f_X}} - \mathbf{K}^{(t)}_{\boldsymbol{f_X}\mathbf{u}}\mathbf{K}^{(t)-1}_{\mathbf{uu}}[\mathbf{K}^{(t)}_{\mathbf{uu}} - \mathbf{S}^{(t)}_{\mathbf{u}}]\mathbf{K}^{(t)-1}_{\mathbf{uu}}\mathbf{K}^{(t)}_{\mathbf{u}\boldsymbol{f_X}})$.

Based on this, we compute the mean of the m-th HiPPO coefficient for $q_t(f)$ as follows,

$$
\begin{aligned}
& E_{q_t(f)}\left[\int f(x)\phi^{(t)}_m(x)dx\right] \\
&= \int E_{q_t(f)}\left[f(x)\right]\phi^{(t)}_m(x)dx \\
&= \left(\int \mathbf{K}^{(t)}_{f_x\mathbf{u}}\mathbf{K}^{(t)-1}_{\mathbf{uu}}\phi^{(t)}_m(x)dx\right)\boldsymbol{m}^{(t)}_{\mathbf{u}} \\
&= \left[\int\int k(x,x')\begin{pmatrix}\phi^{(t)}_1(x')\\ \cdots \\ \phi^{(t)}_M(x')\end{pmatrix}dx'\mathbf{K}^{(t)-1}_{\mathbf{uu}}\phi^{(t)}_m(x)dx\right]\boldsymbol{m}^{(t)}_{\mathbf{u}} \\
&= \left[\int\int k(x,x')\begin{pmatrix}\phi^{(t)}_1(x')\phi^{(t)}_m(x)\\ \cdots \\ \phi^{(t)}_M(x')\phi^{(t)}_m(x)\end{pmatrix}dx'dx\right]\mathbf{K}^{(t)-1}_{\mathbf{uu}}\boldsymbol{m}^{(t)}_{\mathbf{u}} \\
&= \left[\mathbf{K}^{(t)}_{\mathbf{uu}}\right]_{m,:}\mathbf{K}^{(t)-1}_{\mathbf{uu}}\boldsymbol{m}^{(t)}_{\mathbf{u}} \\
&= \left[\boldsymbol{m}^{(t)}_{\mathbf{u}}\right]_m,
\end{aligned}
\tag{C.8}
$$

which is exactly the variational mean of $q(u^{(t)}_m)$. Similarly, the covariance between the l-th and the m-th HiPPO coefficient for $q_t(f)$ can be computed as

$$
\begin{aligned}
& E_{q_t(f)}\left[\left(\int f(x)\phi^{(t)}_l(x)dx\right)\left(\int f(x')\phi^{(t)}_m(x')dx'\right)\right] - \left[\boldsymbol{m}^{(t)}_{\mathbf{u}}\right]_l\left[\boldsymbol{m}^{(t)}_{\mathbf{u}}\right]_m \\
&= \int\int E_{q_t(f)}\left[f(x)f(x')\right]\phi^{(t)}_l(x)\phi^{(t)}_m(x')dxdx' - \left[\boldsymbol{m}^{(t)}_{\mathbf{u}}\right]_l\left[\boldsymbol{m}^{(t)}_{\mathbf{u}}\right]_m \\
&= \int\int \left(k(x,x') - \mathbf{K}^{(t)}_{f_x\mathbf{u}}\mathbf{K}^{(t)-1}_{\mathbf{uu}}[\mathbf{K}^{(t)}_{\mathbf{uu}} - \mathbf{S}^{(t)}_{\mathbf{u}}]\mathbf{K}^{(t)-1}_{\mathbf{uu}}\mathbf{K}^{(t)}_{\mathbf{u}f_{x'}}\right)\phi^{(t)}_l(x)\phi^{(t)}_m(x')dxdx' \\
&\quad + \int\int E_{q_t(f)}\left[f(x)\right]E_{q_t(f)}\left[f(x')\right]\phi^{(t)}_l(x)\phi^{(t)}_m(x')dxdx' - \left[\boldsymbol{m}^{(t)}_{\mathbf{u}}\right]_l\left[\boldsymbol{m}^{(t)}_{\mathbf{u}}\right]_m \\
&= \left[\mathbf{K}^{(t)}_{\mathbf{uu}}\right]_{lm} - \left(\int \mathbf{K}^{(t)}_{f_x\mathbf{u}}\mathbf{K}^{(t)-1}_{\mathbf{uu}}\phi^{(t)}_l(x)dx\right)[\mathbf{K}^{(t)}_{\mathbf{uu}} - \mathbf{S}^{(t)}_{\mathbf{u}}]\left(\int \phi^{(t)}_m(x')\mathbf{K}^{(t)-1}_{\mathbf{uu}}\mathbf{K}^{(t)}_{\mathbf{u}f_{x'}}dx'\right) \\
&= \left[\mathbf{K}^{(t)}_{\mathbf{uu}}\right]_{lm} - \left[\mathbf{K}^{(t)}_{\mathbf{uu}}\right]_{l,:}\mathbf{K}^{(t)-1}_{\mathbf{uu}}\mathbf{K}^{(t)}_{\mathbf{uu}}\mathbf{K}^{(t)-1}_{\mathbf{uu}}\left[\mathbf{K}^{(t)}_{\mathbf{uu}}\right]_{:,m} + \left[\mathbf{K}^{(t)}_{\mathbf{uu}}\right]_{l,:}\mathbf{K}^{(t)-1}_{\mathbf{uu}}\mathbf{S}^{(t)}_{\mathbf{u}}\mathbf{K}^{(t)-1}_{\mathbf{uu}}\left[\mathbf{K}^{(t)}_{\mathbf{uu}}\right]_{:,m} \\
&= \left[\mathbf{S}^{(t)}_{\mathbf{u}}\right]_{lm},
\end{aligned}
\tag{C.9}
$$

which is exactly the variational covariance between $u^{(t)}_l$ and $u^{(t)}_m$ in $q(\boldsymbol{u}^{(t)})$.

Hence, if $f \sim q_t(f)$, then $\int f(x)\phi_m^{(t)}(x)\mathrm{d}x \sim q(u_m^{(t)})$, which implies that we can approximate the posterior OHSVGP with finite basis: $f = \sum_{m=1}^{M} u_m^{(t)} g_m^{(t)}(x)$, $u_m^{(t)} \sim q(u_m^{(t)})$.

C.3 SVGPVAE Model Details

With SVGPVAE, we utilize Jazbec et al. (2021) and notation from Zhu et al. (2023), and we have the following encoder-decoder model:

$$p(\mathbf{y}_{1:T}) = p(\mathbf{f}_{1:T}) \prod_{t=1}^{T} p(\mathbf{y}_t|\mathbf{f}_t), \tag{C.10}$$

with likelihood $p(\mathbf{y}_t|\mathbf{f}_t) = \mathcal{N}(\mathbf{y}_t|\varphi(\mathbf{f}_t), \sigma^2 I)$ and decoder network $\varphi : \mathbb{R}^L \to \mathbb{R}^{d_y}$. The encoder $\phi : \mathbb{R}^{d_y} \to \mathbb{R}^{2L}$ yields $(\tilde{\mathbf{y}}_t^{1:L}, \tilde{\mathbf{v}}_t{}^{1:L}) = \phi(\mathbf{y}_t)$. $\mathbf{f}_t$ follows an L-dimensional multi-output GP and its approximate posterior is given by:

$$q(\mathbf{f}_{1:T}) = \prod_{l=1}^{L} p(\mathbf{f}_{1:T}^l|\mathbf{u}_m^l) q(\mathbf{u}^l), \quad q(\mathbf{u}^l) = \mathcal{N}(\mathbf{u}^l|\mathbf{m}^l, \mathbf{A}^l), \tag{C.11}$$

$$\mathbf{S}^l = \mathbf{K}_{\mathbf{uu}}^l + \mathbf{K}_{\mathbf{uf}}^l \mathrm{diag}(\tilde{\boldsymbol{v}}_{1:T}^l)^{-1} \mathbf{K}_{\mathbf{fu}}^l, \quad \mathbf{m}^l = \mathbf{K}_{\mathbf{uu}}^l (\mathbf{S}^l)^{-1} \mathbf{K}_{\mathbf{uf}}^l \mathrm{diag}(\tilde{\boldsymbol{v}}_{1:T}^l)^{-1} \tilde{\mathbf{y}}_{1:T}^l, \tag{C.12}$$

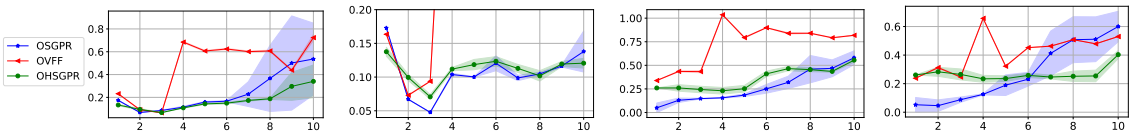
$$\mathbf{A}^l = \mathbf{K}_{\mathbf{uu}}^l (\mathbf{S}^l)^{-1} \mathbf{K}_{\mathbf{uu}}^l, \tag{C.13}$$

where $p(\mathbf{f}_{1:T}^l|\mathbf{u}_m^l)$ is the prior conditional distribution.

Following Jazbec et al. (2021), the objective function is defined as:

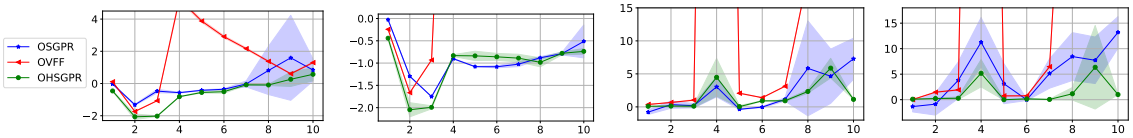
$$\mathcal{L}_{\text{SVGPVAE}}(\theta) = \sum_{t=1}^{T} \Big[\mathbb{E}_{q(\mathbf{f}_t)} \log p(\mathbf{y}_t|\mathbf{f}_t) - \log \mathcal{N}(\mathbf{f}_t|\tilde{\mathbf{y}}_t, \tilde{\mathbf{v}}_t) \Big] + \sum_{l=1}^{L} \mathcal{L}_H^l, \tag{C.14}$$

where $\mathcal{L}_H^l$ is the "Hensman" ELBO described in Equation 7 of Jazbec et al. (2021). Since the variational parameters $\mathbf{m}^l$ and $\mathbf{S}^l$, and the likelihood are all amortized by neural networks, we further add EWC (Elastic Weight Consolidation; (Kirkpatrick et al., 2017)) regularization for both encoder and decoder networks to the loss above for continual learning.



(a) Solar, M=50 (b) Solar, M=150 (c) Audio, M=100 (d) Audio, M=200

Figure C.1: Test set RMSE over the learned tasks vs. number of learned tasks for Solar Irradiance and Audio signal prediction dataset (keep updating kernel hyperparameters).



(a) Solar, M=50 (b) Solar, M=150 (c) Audio, M=100 (d) Audio, M=200

Figure C.2: Test set NLPD over the learned tasks vs. number of learned tasks for Solar Irradiance and Audio signal prediction dataset (keep updating kernel hyperparameters).

C.4 Additional Results

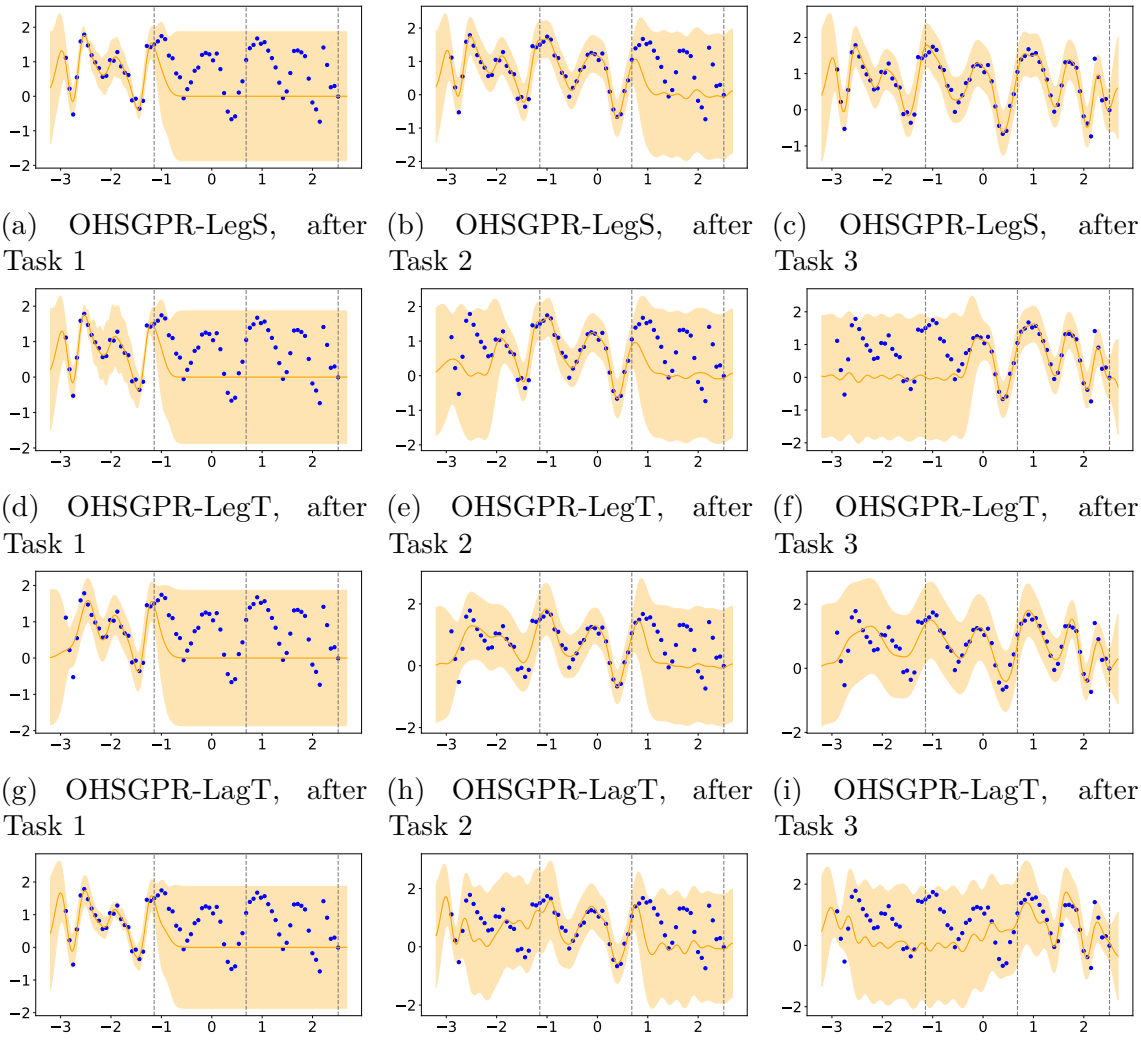
C.4.1 Results for Time Series Regression with Trainable Kernel Hyperparameters

Figure C.1 and C.2 show RMSE and NLPD results for time series regression experiments based on trainable kernel hyperparameters (i.e., keep optimizing kernel hyperparameters online in all the tasks). Notice that OVC is only compatible with fixed kernel (Maddox et al., 2021), so we don't consider it here. Compared with the results based on fixed kernel in the main text, here all methods show less stable performance. Previous works either find a well-performed fixed kernel (Maddox et al., 2021) or consider generalized VI by scaling the KL terms in the online ELBO objective with a positive factor requiring careful tuning to mitigate the unstable online optimization of kernel hyperparameters (Stanton et al., 2021; Kapoor et al., 2021).

C.4.2 Comparison of Basis–measure Variants

Figure C.3 shows the results of OHSGPR applied to a toy time-series regression dataset, where the data is split chronologically into three equal segments, used as Tasks 1–3 in an online learning setup. The figure compares the effect of several variants of the HiPPO operators (Gu et al., 2020, 2023) when used for OHSGPR. Subfigures (a–c) correspond to HiPPO-LegS as used in all of our main experiments. Subfigures (d–f) apply HiPPO-LegT, (g–i) apply HiPPO-LagT, based on the Laguerre

polynomial basis, and (j–l) apply HiPPO-FouT, based on Fourier basis functions. The detailed formulation of these HiPPO variants can be found in Section 2.1.2 and Appendix A.1. While OHSGPR-LegS successfully memorizes all the past tasks, OHSGPR-LegT, OHSGPR-LagT and OHSGPR-FouT all demonstrate catastrophic forgetting to certain degree since instead of using the uniform measure over the past (as is used in HiPPO-LegS), they are based on measures which place more mass over the recent history. LegT and FouT use a fixed-length sliding window measure, while LagT uses exponentially decaying measure, which assigns more importance to recent history.



(a) OHSGPR-LegS, after Task 1
(b) OHSGPR-LegS, after Task 2
(c) OHSGPR-LegS, after Task 3
(d) OHSGPR-LegT, after Task 1
(e) OHSGPR-LegT, after Task 2
(f) OHSGPR-LegT, after Task 3
(g) OHSGPR-LagT, after Task 1
(h) OHSGPR-LagT, after Task 2
(i) OHSGPR-LagT, after Task 3
(j) OHSGPR-FouT, after Task 1
(k) OHSGPR-FouT, after Task 2
(l) OHSGPR-FouT, after Task 3

Figure C.3: Comparison of OHSGPR based on different HiPPO variants on a toy online regression dataset.

Appendix D

Supplementary Material for Chapter 5

D.1 Proof of Theorem 5.4.1

Proof. To derive $\hat{\boldsymbol{z}}_T^{(g)*}(\boldsymbol{z}_{s_1}, \cdots, \boldsymbol{z}_{s_k}) = \arg\min_{\hat{\boldsymbol{z}}_T^{(g)}} \mathbb{E}_{p(\boldsymbol{z}_T, \boldsymbol{z}_{s_1}, \cdots, \boldsymbol{z}_{s_k})}[||\boldsymbol{z}_T - \hat{\boldsymbol{z}}_T^{(g)}||_2^2]$, we first compute the gradient,

$$\begin{aligned}\nabla_{\hat{\boldsymbol{z}}_T^{(g)}} \mathbb{E}_{p(\boldsymbol{z}_T, \boldsymbol{z}_{s_1}, \cdots, \boldsymbol{z}_{s_k})}[||\boldsymbol{z}_T - \hat{\boldsymbol{z}}_T^{(g)}||_2^2] &= \mathbb{E}_{p(\boldsymbol{z}_{s_1}, \cdots, \boldsymbol{z}_{s_k})}\{\nabla_{\hat{\boldsymbol{z}}_T^{(g)}} \mathbb{E}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{s_1}, \cdots, \boldsymbol{z}_{s_k})}[||\boldsymbol{z}_T - \hat{\boldsymbol{z}}_T^{(g)}||_2^2]\} \\ &= 2\mathbb{E}_{p(\boldsymbol{z}_{s_1}, \cdots, \boldsymbol{z}_{s_k})}\{\mathbb{E}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{s_1}, \cdots, \boldsymbol{z}_{s_k})}[\boldsymbol{z}_T - \hat{\boldsymbol{z}}_T^{(g)}]\}.\end{aligned} \tag{D.1}$$

Setting the above gradient to 0 gives us

$$\mathbb{E}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{s_1}, \cdots, \boldsymbol{z}_{s_k})}[\boldsymbol{z}_T - \hat{\boldsymbol{z}}_T^{(g)}] = 0 \implies \hat{\boldsymbol{z}}_T^{(g)*}(\boldsymbol{z}_{s_1}, \cdots, \boldsymbol{z}_{s_k}) = \mathbb{E}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{s_1}, \cdots, \boldsymbol{z}_{s_k})}[\boldsymbol{z}_T]. \tag{D.2}$$

The minimum reconstruction error is obtained by plugging $\hat{\boldsymbol{z}}_T^{(g)*}(\boldsymbol{z}_{s_1}, \cdots, \boldsymbol{z}_{s_k})$ in $\mathbb{E}_{p(\boldsymbol{z}_T, \boldsymbol{z}_{s_1}, \cdots, \boldsymbol{z}_{s_k})}[||\boldsymbol{z}_T - \hat{\boldsymbol{z}}_T^{(g)}||_2^2]$,

$$\begin{aligned}\min_{\hat{\boldsymbol{z}}_T^{(g)}} \mathbb{E}_{p(\boldsymbol{z}_T, \boldsymbol{z}_{s_1}, \cdots, \boldsymbol{z}_{s_k})}[||\boldsymbol{z}_T - \hat{\boldsymbol{z}}_T^{(g)}||_2^2] &= \mathbb{E}_{p(\boldsymbol{z}_T, \boldsymbol{z}_{s_1}, \cdots, \boldsymbol{z}_{s_k})}[||\boldsymbol{z}_T - \mathbb{E}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{s_1}, \cdots, \boldsymbol{z}_{s_k})}[\boldsymbol{z}_T]||_2^2] \\ &= \mathbb{E}_{p(\boldsymbol{z}_{s_1}, \cdots, \boldsymbol{z}_{s_k})}[\mathrm{Var}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{s_1}, \cdots, \boldsymbol{z}_{s_k})}(\boldsymbol{z}_T)].\end{aligned} \tag{D.3}$$

Similarly,

$$\begin{aligned}\min_{\hat{\boldsymbol{z}}_T^{(h)}} \mathbb{E}_{p(\boldsymbol{z}_T,\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_l})}[||\boldsymbol{z}_T-\hat{\boldsymbol{z}}_T^{(h)}||_2^2] &= \mathbb{E}_{p(\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_l})}[\mathrm{Var}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_l})}(\boldsymbol{z}_T)],\\ \hat{\boldsymbol{z}}_T^{(h)*}(\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_l}) &= \mathbb{E}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_l})}[\boldsymbol{z}_T],\end{aligned} \tag{D.4}$$

We now show that the reconstruction error can never increase with more previous frames as inputs by observing that with $T > s_1 > \cdots > s_k > \cdots > s_l$,

$$\begin{aligned}&\min_{\hat{\boldsymbol{z}}_T^{(h)}}\mathbb{E}_{p(\boldsymbol{z}_T,\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_l})}[||\boldsymbol{z}_T-\hat{\boldsymbol{z}}_T^{(h)}||_2^2] = \mathbb{E}_{p(\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_l})}[\mathrm{Var}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_l})}(\boldsymbol{z}_T)]\\ &\quad\le \mathbb{E}_{p(\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_k})}[\mathrm{Var}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_k})}(\boldsymbol{z}_T)] = \min_{\hat{\boldsymbol{z}}_T^{(g)}} \mathbb{E}_{p(\boldsymbol{z}_T,\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_k})}[||\boldsymbol{z}_T-\hat{\boldsymbol{z}}_T^{(g)}||_2^2].\end{aligned} \tag{D.5}$$

Indeed,

$$\begin{aligned}&\mathbb{E}_{p(\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_l})}[\mathrm{Var}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_l})}(\boldsymbol{z}_T)] - \mathbb{E}_{p(\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_k})}[\mathrm{Var}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_k})}(\boldsymbol{z}_T)]\\ &= \mathbb{E}_{p(\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_k})}\{\mathbb{E}_{p(\boldsymbol{z}_{s_{k+1}},\cdots,\boldsymbol{z}_{s_l}|\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_k})}[\mathrm{Var}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_l})}(\boldsymbol{z}_T)] - \mathrm{Var}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_k})}(\boldsymbol{z}_T)\}\\ &= -\mathbb{E}_{p(\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_k})}\{\mathrm{Var}_{p(\boldsymbol{z}_{s_{k+1}}\cdots,\boldsymbol{z}_{s_l}|\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_k})}(\mathbb{E}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_l})}[\boldsymbol{z}_T])\} \qquad \text{(Law of total variance)}\\ &\le 0.\end{aligned} \tag{D.6}$$

Notice that if $\boldsymbol{z}_T|\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_k} \overset{d}{=} \boldsymbol{z}_T|\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_l}$, then the above difference will become 0:

$$\begin{aligned}&\mathrm{Var}_{p(\boldsymbol{z}_{s_{k+1}}\cdots,\boldsymbol{z}_{s_l}|\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_k})}(\mathbb{E}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_l})}[\boldsymbol{z}_T])\\ &= \mathrm{Var}_{p(\boldsymbol{z}_{s_{k+1}}\cdots,\boldsymbol{z}_{s_l}|\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_k})}(\mathbb{E}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_k})}[\boldsymbol{z}_T])\\ &= 0,\end{aligned} \tag{D.7}$$

since $\mathbb{E}_{p(\boldsymbol{z}_T|\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_k})}[\boldsymbol{z}_T]$ is a function of $\boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_k}$ only.

Therefore, for strict inequality, it is necessary to avoid $\boldsymbol{z}_T \perp\!\!\!\perp \boldsymbol{z}_{s_{k+1}},\cdots,\boldsymbol{z}_{s_l} \mid \boldsymbol{z}_{s_1},\cdots,\boldsymbol{z}_{s_k}$, which includes first-order Markov chain ($\boldsymbol{z}_T \to \cdots \boldsymbol{z}_{s_k} \to \boldsymbol{z}_{s_l}$) as a special case.

□

D.2 Hyperparameters

During our experiments, we first choose the largest number of frames to be the largest one that we can train on our GPU (an NVIDIA RTX A6000) and they are 18 and 8 for CViViT and Video Diffusion, respectively. Then we consider including

the results with the number of frames approximately being half of the maximal one (8 and 4 for CViViT and Video Diffusion, respectively). The hyperparameters for our experiments are shown in the tables below.

Table D.1: Hyperparamters used for C-ViViT architecture and optimizer.

	1-frame	8-frame	18-frame
Number of spatial layers	8	4	4
Number of temporal layers	-	4	4
Embedding dimension	512	512	512
Hidden dimension	512	512	512
Number of heads	8	8	8
Learning rate	1e-4	1e-4	1e-4
Learning rate scheduler	Cosine decay	Cosine decay	Cosine decay
Number of training steps	100k	100k	100k
Batch size	64	64	64

Table D.2: Hyperparamters used for VideoGPT architecture and optimizer.

	1-frame	8-frame	18-frame
Number of layers	8	8	8
Embedding dimension	144	144	144
Hidden dimension	144	144	144
Number of heads	4	4	4
Learning rate	1e-4	1e-4	1e-4
Learning rate scheduler	Cosine decay	Cosine decay	Cosine decay
Number of training steps	100k	100k	100k
Batch size	64	64	64

Table D.3: Hyperparamters used for Phenaki architecture and optimizer.

	1-frame	8-frame	18-frame
Number of layers	6	6	6
Embedding dimension	512	512	512
Hidden dimension	512	512	512
Number of heads	8	8	8
Learning rate	1e-4	1e-4	1e-4
Learning rate scheduler	Cosine decay	Cosine decay	Cosine decay
Number of training steps	200k	200k	200k
Batch size	64	64	64

Table D.4: Hyperparamters used for UNet architecture and optimizer for Video diffusion.

	1-frame	4-frame	8-frame
Number of downsampling/upsampling layers	4	4	4
Number of residual blocks	2	2	2
Base channel size	128	128	128
Number of diffusion steps per generating a frame	1000	1000	1000
Learning rate	1e-4	1e-4	1e-4
Number of training steps	500k	500k	500k
Batch size	32	32	32