Gaussian Process Surrogate Models for Neural Networks

Ishita Agarwal, Rachit Khamesra, Ayush Anand, Chinmay Pillai *
IIT Kanpur

Abstract

Deep learning systems, while central to modern machine learning, often operate as opaque "black-boxes" where internal mechanisms remain concealed, complicating the identification of performance issues in real-world applications. Leveraging Gaussian Processes (GPs) as surrogate models provides a theoretical bridge linking neural networks to GPs through the conceptualization of infinitely wide networks capable of modeling complex non-linearities in data. Our project focuses on utilizing GPs to predict leave-one-out (LOO) probabilities and assess the significance of data points in large datasets. The goal is to enhance the efficiency of neural network training and facilitate real-time updates, thereby improving both the effectiveness and interpretability of deep learning models in practical settings. Additionally, we explore variational techniques, particularly those developed by Titsias, to address computational complexities and enhance the accuracy of surrogate model estimates.

1 Introduction

"Gaussian Process Surrogate Models for Neural Networks" presents an innovative approach to understanding and predicting the behavior of deep learning systems through the construction of Gaussian Process (GP) surrogate models. By diverging from the traditional method of deriving kernels for infinite neural networks, this study focuses on learning kernels empirically from the observable behavior of finite neural networks. Utilizing these models, we can address practical challenges such as identifying influential data points. The approach underscores the utility of GPs in achieving a deeper understanding of NN behaviors in a theoretically sound and practically applicable framework.

Surrogate modeling involves approximating complex functions with simpler models that emulate their behavior, widely used in optimization to query expensive functions and in analyzing large physical systems. Gaussian Processes (GPs) are a powerful statistical modeling technique used primarily for regression and probabilistic classification tasks within the field of machine learning. The fundamental idea behind GPs is to treat predictions as distributions rather than as single point estimates, which allows for a natural quantification of uncertainty in predictions. GPs define a prior over functions, where any finite collection of function values (associated with different inputs) is assumed to be jointly Gaussian distributed. The essence of a GP lies in its mean and covariance functions. The mean function, often set to zero for simplicity, represents the expected output over the input space. The covariance function, or kernel, is more critical, defining the relationship between different inputs. This function specifies how outputs are correlated based on their inputs, allowing the GP to encode assumptions about the function's behavior, such as smoothness or periodicity. Gaussian Processes are highly flexible and can adapt to a wide range of functions thanks to the choice of kernel. The interpretability and the explicit uncertainty modeling make GPs particularly appealing in fields that require robust predictions accompanied by reliable confidence measures.

*CS772: Probabilistic Machine Learning

Mathematically, a GP is expressed as:

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')), \tag{1}$$

where $m(x)$ is the mean function and $k(x, x')$ is the kernel function. The kernel function, such as the squared exponential:

$$k(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2l^2}\right), \tag{2}$$

plays a critical role in the GP's ability to model functions, where σ^2 is the variance and l is the length-scale parameter, both of which control the smoothness and variability of the function predictions.

The flexibility of Gaussian Processes stems from their non-parametric nature, allowing them to adapt their complexity to the amount of data available, and from their ability to provide predictive distributions rather than single point estimates. This property is particularly valuable for quantifying uncertainty in predictions, which is crucial for many applications such as decision-making under uncertainty and exploration-exploitation tasks in reinforcement learning.

2 Connecting Gaussian Process Surrogate Models with Neural Tangent Kernels

2.1 Introduction

The integration of Gaussian Process (GP) surrogate models with Neural Tangent Kernels (NTKs) offers a novel perspective on the study of neural networks. While NTKs provide a rigorous theoretical foundation by exploring the behavior of neural networks in the limit of infinite width, GP surrogate models address the complexities of real-world, finite neural networks. This section examines the theoretical underpinnings of NTKs, the empirical motivations behind GP surrogate models, and their practical implications in neural network research.

2.2 Theoretical Background of Neural Tangent Kernels

Neural Tangent Kernels (NTKs) represent a pivotal concept in the theory of deep learning, particularly concerning infinitely wide networks. These kernels arise from considering the gradient dynamics during training, where the network's width approaches infinity, and the kernel governing the training dynamics, the NTK, remains constant. Mathematically, for a neural network function $f(\mathbf{x}; \theta)$ parameterized by θ, the NTK $\Theta(\mathbf{x}, \mathbf{x}')$ is defined as:

$$\Theta(\mathbf{x}, \mathbf{x}') = \mathbb{E}_\theta \left[\nabla_\theta f(\mathbf{x}; \theta) \cdot \nabla_\theta f(\mathbf{x}'; \theta)\right], \tag{3}$$

This kernel quantifies how infinitesimal changes in the network parameters affect the outputs, averaged over all possible initializations, underpinning a connection to Gaussian Processes.

2.3 Empirical Motivations for Gaussian Process Surrogate Models

While NTKs provide a clean, theoretical lens to view the training of neural networks, they inherently rely on the assumption of infinite width, which is impractical for real-world applications. This gap between theory and practice motivates the development of GP surrogate models, which are designed to empirically learn from the observable behavior of finite networks. These models focus on capturing:

- **Spectral biases:** Phenomena where networks show a predilection for learning certain frequencies of inputs more efficiently than others.
- **Initialization pathologies:** These refer to issues that arise due to specific initial conditions of network parameters, which can significantly impact learning dynamics and model performance.

Such empirical insights are critical for developing models that reflect the true behavior and challenges encountered in practical settings. By modeling networks through GPs, researchers can predict the effect of architectural changes or training regimen modifications without the need for exhaustive retraining.

3 Learning a Gaussian Process Surrogate Model from Neural Network Predictions

3.1 Overview

This section describes the methodology for developing a Gaussian Process (GP) surrogate model that utilizes predictions from neural networks across various initializations and datasets. The process includes estimating GP kernel hyperparameters by maximizing the marginal likelihood, considering multiple pairs of neural network configurations and datasets.

3.2 Formal Framework

The primary objective of this framework is to encapsulate common characteristics among a family of neural network models, denoted as $\mathcal{F}$, when applied to a collection of datasets, denoted as $\mathcal{D}$. This is particularly focused on cases where the datasets may not be directly used for training the neural networks; for example, scenarios involving untrained networks where $\mathcal{D}$ is empty.

- **Model Family** $\mathcal{F}$: Consists of a collection of neural network models $\{g_0, \ldots, g_R\}$ that share common design elements such as architecture and training procedures but vary in aspects that are randomized before or during training, such as parameter initializations.
- **Dataset Family** $\mathcal{D}$: Comprises datasets $\{D_0, \ldots, D_S\}$ that share some structural characteristics, reminiscent of those found in multi-task and meta-learning contexts.

In this supervised learning setting, each dataset in $\mathcal{D}$ comprises input-target pairs (X, y). Importantly, the parameters θ of the surrogate model are fitted not directly to the datasets themselves but to a 'behavioral dataset'. This dataset is constructed from the outputs of the neural network models as they are evaluated on the dataset family, capturing the networks' predictions at specific test inputs.

3.3 Constructing the Surrogate Model Training Dataset

The construction of the surrogate model training dataset is executed through the following methodical steps:

1. **Sampling Model and Dataset Indices:** Initially, a model index r and a dataset index s are randomly selected. This sampling determines the model from $\mathcal{F}$ and the dataset from $\mathcal{D}$ to be used in this iteration.
2. **Splitting the Dataset:** The chosen dataset D_s is divided into a training set D_{train} and an evaluation set D_{eval}. This split enables the model to be trained on one subset of data and validated on another to assess performance.
3. **Model Training:** The neural network model g_r is trained on the training set $D_{\text{train}} = (X_{\text{train}}, y_{\text{train}})$. The training adheres to the procedures specified by $\mathcal{F}$, including predefined settings like architecture, training methods, and random initialization schemes.
4. **Collecting Predictions:** Post-training, the fitted model g_{fit} is utilized to predict outcomes on the evaluation set. The predictions $g_{\text{fit}}(X_{\text{eval}})$ are collected, pairing the evaluation inputs X_{eval} with the predicted outputs from the trained model.
5. **Aggregating Data:** Inputs from the evaluation set and corresponding predictions by the neural network are aggregated. This forms pairs of ground truth inputs and neural network predictions, which serve as the behavioral targets. These pairs are compiled across various model and dataset indices to create the comprehensive surrogate model training dataset.

The framework outlined employs Gaussian Processes (GPs) as surrogate models to encapsulate and scrutinize the inductive biases intrinsic to families of neural network (NN) models, which are pivotal for their extrapolation capabilities. GPs are chosen due to their flexibility and interpretability, allowing insights into the data-driven characteristics that influence model behaviors. The choice is also supported by established theoretical connections between GPs and NNs, demonstrating that as the width of a neural network, such as a multilayer perceptron (MLP), increases, its functional representation converges to that of a GP. This relationship has been further validated by a series of

studies from Neal (1996) to more recent works in 2019, which highlight that the implicit priors in NN designs can be explicitly modeled using GPs. Diverging from theoretical approaches that derive analytical kernels from models of NNs, this framework adopts an empirical method where GP kernels are learned directly from the outputs of finite, operational NNs.

A series of experiments were conducted to elucidate the capabilities of Gaussian Process (GP) surrogate models in capturing established neural network (NN) phenomena and in demonstrating their applicability in practical machine learning scenarios.

4 Experiment: Amortized Influence Estimation

The major concern for models in the current scenario is the sheer amount of data available for training but it also estimated that majority of the data is non-influential for models i.e. models already perform well on that kind of data. The premise of the experiment revolves around estimation of the influence/importance of data points to the accuracy of the model.

As explained in the base paper, the primary computational bottleneck of fitting a GP is the O(N^3) cost of Gram matrix inversion. However, once this matrix is inverted, certain computations that seem costly can actually be performed cheaply. For example, the leave one-out predictive (LOO) distributions can be computed analytically [Rasmussen and Williams, 2006]. The LOO distributions characterize how GP predictions change after removing a training point. In particular, we use the LOO predictive distributions of a surrogate GP to detect influential points for NNs with Fourier feature layers. We try to test results from the base paper to show that the surrogate LOO predictions are consistent with NN LOO predictions obtained from costly retraining.

4.1 Implementation

1. Build a neural network with the specified architecture- NN with 4 hidden layers of 1024 units, ReLU activations.
2. The inputs to the NNs are passed through a single Fourier feature layer with parameter p that controls the spectral bias of the NN. p is set to 1 and Fourier feature mapping is implemented with frequencies ranging from 1 to 4. This feature engineering step converts our input size from (n,1) to (n,4), where 'n' is the number of training data points.
3. We train the network using full-batch gradient descent with stochastic gradient descent (SGD) and a learning rate of $\eta = 10^{-3}$ for 2000 iterations.
4. We consider a target function $f(x) = sin(30(x-0.95)^4)cos(2(x-0.95)) + (x-0.95)/2$ This target function is non-stationary, with higher frequency behavior in the left half of its input domain and lower frequency behavior in the right.
5. A set of 100 periodic samples between 0 to 1 and their corresponding mappings in the target function acts as the training data available for the models.
6. We then perform a 100-fold Leave One Out Cross Validation (LOOCV) step and store the predictions on the left out point for each of the 100 models.
7. The LOO error is computed as the difference between the NN LOO predictions and the predictions of the NN trained on the full dataset. This error acts as a measure of the importance each data point has on training. Higher error corresponds to more impact in the model training.
8. We fit a GP with an RBF kernel over the test data $(x_{data}, NN(x_{data}))$ and as per the paper.
9. Following this, another 100-fold LOOCV step is performed on the data but now using GP models. Each of the 100 models provide us with the predictive mean μ_i and the predictive standard deviation σ_i.
10. The GP LOO predictive probability for each of the 100 LOO GP models leaving out training sample *i* is calculated as:
    $log(y_i|X_{-i}, y_{-i}) = -\frac{1}{2}log\sigma_i^2 - \frac{(y_i-\mu_i)^2}{2\sigma_i^2} - \frac{1}{2}log2\pi$.
11. Post this estimation, we compare the surrogate GP LOO and NN LOO predictions for points. And then, we plot a scatter for the predictions, and check the Spearman's rank correlation.

12. The code we used to implement this can be found at: *GP Surrogate for Neural Networks*

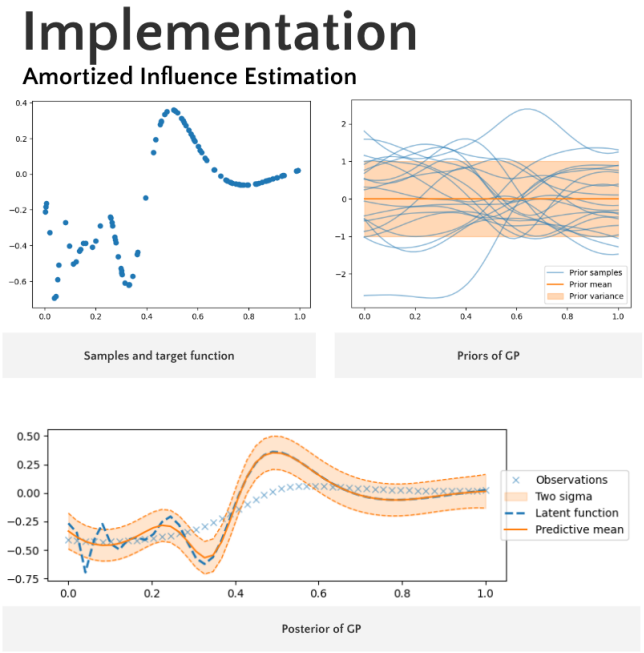

Figure 1: Top-left : Scatter plot for data, Top-right : Prior function for GP, Bottom : Posterior functions for GP

4.2 Experimental results

Indeed, we observe a positive Spearman's rank correlation and hence the GP LOO predictions at the influential points are somewhat predictive of the NN LOO prediction. In the Figure, we've plotted the (negative) LOO log probability at each input point against the actual LOO error at the training input.

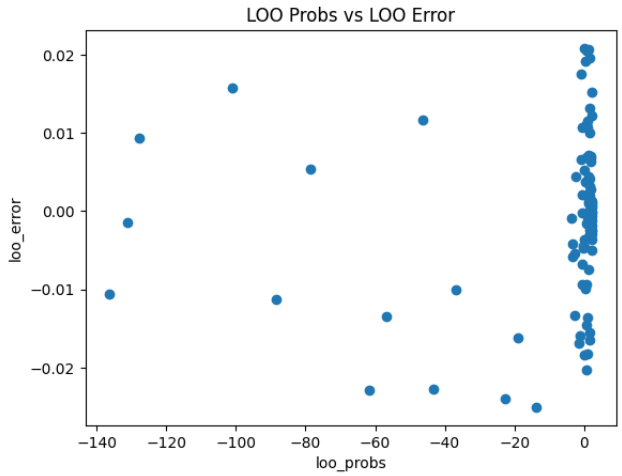

Figure 2: Scatter plot for GP-LOO and NN-LOO

One of the key differences from the paper is the zero jitter (error) added to the NNs predictions while training the GP. The Spearman's rank correlation coefficient ρ shows a slightly positive relationship between the negative LOO log probability from the GP, and the NN LOO error. More negative log

probabilities indicate the importance of those points, while the cluster at 0 log probability indicates that the neural network has learnt the function well and these points are not crucial to the network's training.

This helps understand that many times, the amount of training data being fed into neural network is not required as the network has already generalised well over that area of the function and here, the importance of designing acquisition functions for efficiency gains is important.

5 Fourier Feature Layers in Neural Networks

Fourier feature layers, also known as Fourier feature mappings, are a transformative technique in machine learning designed to enhance the capability of neural networks, particularly multilayer perceptrons (MLPs), to learn high-frequency functions in low-dimensional input spaces. This method is highly relevant in fields such as computer vision and graphics, where the representation and manipulation of complex spatial information are crucial.

5.1 Importance of Fourier Feature Layers

The primary significance of Fourier feature layers lies in their ability to overcome the spectral bias commonly observed in neural networks, especially MLPs. Spectral bias refers to the tendency of these networks to preferentially learn lower-frequency functions, which can be a limitation in tasks requiring the capture of high-frequency details. By integrating Fourier feature layers, networks are endowed with the capability to also effectively learn higher-frequency features. This is particularly critical in applications involving low-dimensional input spaces, such as 2D or 3D coordinates, where standard MLPs often struggle to learn complex functions that represent textures or detailed spatial variations. Furthermore, in the domain of computer graphics, employing Fourier features allows MLPs to represent complex 3D objects and scenes more compactly and efficiently than traditional discrete representations like voxel grids or meshes.

5.2 Technical Explanation

Fourier feature mappings enhance the input representations by applying a sinusoidal transformation to the input coordinates before they enter the MLP. This transformation is mathematically expressed as:

$$\gamma(v) = \begin{bmatrix} \cos(2\pi Bv) \\ \sin(2\pi Bv) \end{bmatrix}^T, \tag{4}$$

where v represents the input coordinates and B is a matrix containing parameters that determine the frequencies of the sinusoids involved. This matrix B can be learned or predetermined based on the specific requirements of the task at hand. The sinusoidal transformation effectively alters the way the MLP interprets the input space, enabling it to model higher-frequency functions that are otherwise challenging to capture with standard architectural configurations.

5.3 Practical Benefits

Incorporating Fourier feature layers into neural networks has demonstrated substantial improvements in several tasks. Networks equipped with Fourier feature mappings show enhanced performance due to their increased capacity to capture and represent variations at different scales and frequencies. In our experiments, we realised that the training loss decreased from 0.0398 to 0.0249 with Fourier feature layer which is an improvement of approximately 40%.

Moreover, the adaptability of Fourier feature mappings provides an additional advantage. By tuning parameters such as the frequency and amplitude of the sinusoids, these mappings can be customized to optimize performance for specific applications. This flexibility makes Fourier feature layers a robust solution to the challenge of learning high-frequency details in low-dimensional domains, thereby serving as a critical tool in advancing the capabilities of machine learning systems in the fields of vision and graphics.

6 Variational Model Selection

Gaussian Processes (GPs) are renowned for their ability to provide not only predictions but also uncertainty estimates, GPs are inherently Bayesian, capturing our beliefs about the data in a non-parametric manner. However, despite their elegance, GPs face a critical limitation: computational complexity. The traditional implementation of GPs scales cubically with the number of data points, rendering them impractical for large datasets. This computational bottleneck compels us to seek more scalable solutions without significantly compromising the model's expressiveness or predictive power.

In pursuit of a more computationally tractable approach, the paper "Variational Model Selection for Sparse Gaussian Process Regression" introduces a sparse variant of Gaussian Processes that tackles scalability issues directly. Titsias proposes a variational framework that treats the selection of inducing variables and kernel hyperparameters as a model optimization problem. The core of this methodology involves:

6.1 Variational Bound Maximization

Instead of the traditional approach of modifying the likelihood function or the GP prior, this method maximizes a lower bound to the exact log marginal likelihood. This maximization involves selecting inducing inputs and hyperparameters that minimize the Kullback-Leibler divergence between the variational distribution and the exact posterior over the latent functions. Inducing Inputs as Variational Parameters: The inducing inputs are treated not just as fixed parameters but as variational parameters that are optimized during the learning process. This approach allows for the dynamic adjustment of these inputs based on the data, which leads to better model flexibility and accuracy. By applying this methodology to regression tasks, it enables the handling of large datasets by effectively reducing the computational complexity from $O(N^3)$ to $O(NM^2)$, where M is much smaller than N. The variational framework introduced by Titsias is used primarily for the following reasons:

1. Scalability: By reducing the computational demands, this method allows Gaussian Processes to scale to large datasets, which is crucial for applications involving extensive data such as in deep learning.
2. Improved Hyperparameter Estimation: The precise selection of hyperparameters and inducing inputs enhances the model's ability to generalize from training data to unseen data, thereby improving prediction accuracy.
3. Robustness Against Overfitting: The method's focus on minimizing divergence ensures that the model does not overfit to the training data, maintaining robustness and reliability in predictions.

7 Future work

In the context of "Gaussian Process Surrogate Models for Neural Networks," Titsias's methodology can significantly enhance the surrogate models used for regression tasks:

1. Enhanced Model Accuracy and Efficiency: By integrating this variational approach, the surrogate models can achieve higher accuracy in predicting neural network behaviors by closely approximating the underlying models with a more precise estimation of the GP parameters.
2. Application in Neural Network Analysis: Surrogate models often need to replicate the function of complex neural networks in a more computationally feasible manner. Titsias's method allows these models to efficiently handle the complexities involved while maintaining high accuracy and speed, making them particularly useful for iterative tasks like hyperparameter tuning in neural networks.
3. Theoretical and Practical Synergy: The theoretical robustness provided by the variational framework ensures that the surrogate models are not only practically efficient but also grounded in a rigorous statistical foundation, enhancing their utility in research and practical applications.

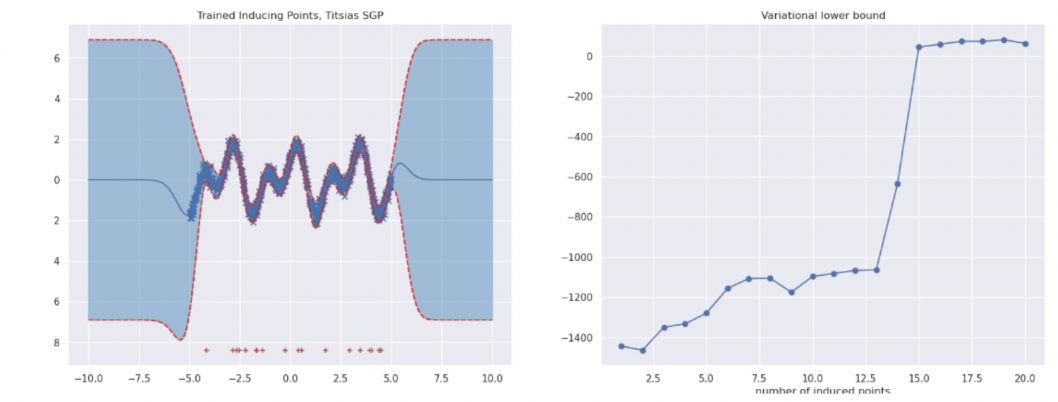

Figure 3: **Left**: Plot showcases the sparse GP's ability to use a limited number of data points, thus overcoming the computational limitations of traditional GPs. **Right**: The variational lower bound as a function of the number of inducing points; highlights how the addition of inducing points initially leads to significant gains in model performance. And, as the model reaches a certain complexity, these improvements plateau.

While we have successfully implemented Titsias's sparse GP methodology, we have yet to fully explore its integration within our surrogate models. This integration represents a promising future direction for our research, potentially enhancing our models' complexity management and estimation capabilities. Our future experiments will delve into this integration, aiming to harness the full potential of Titsias's approach to elevate the performance and scalability of our surrogate models further.

8 Contribution

1. Rachit Khamesra (200747) : Code for Neural Network, Report, Presentation
2. Chinmay Pillai (200298): Code for GP, Neural Network
3. Ishita Agarwal (200458): Titsias' paper integration, Report, Presentation
4. Ayush Anand (200238): Presentation

References

[1] Li, Michael & Grant, Erin & Griffiths, Thomas. (2022). *Gaussian process surrogate models for neural networks*. 10.48550/arXiv.2208.06028.

[2] Titsias, Michalis. (2009). *Variational Model Selection for Sparse Gaussian Process Regression.*

[3] Tancik, Matthew & Srinivasan, Pratul & Mildenhall, Ben & Fridovich-Keil, Sara & Raghavan, Nithin & Singhal, Utkarsh & Ramamoorthi, Ravi & Barron, Jonathan & Ng, Ren. (2020). *Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains.* NeurIPS 2020

[3] Thomas Pinder and Daniel Dodd. *GPJax: A Gaussian Process Framework in JAX* 2022

[4] Lilian Weng. *Some Math behind Neural Tangent Kernel* https://lilianweng.github.io/posts/2022-09-08-ntk/

[5] C. E. Rasmussen and C. K. Williams. *Gaussian processes for machine learning*. Jan. 2006.

[6] Jack Dabrowski, Maria Janicka Lucasz Sienkiewicz *Fourier Feature Encoding*. Aug. 2022.