Scalable Gaussian Processes, with Guarantees: Kernel Approximations and Deep Feature Extraction

Constantinos Daskalakis
Department of Electrical Engineering
and Computer Science, MIT
costis@csail.mit.edu

Petros Dellaportas
Department of Statistical Science,
University of College London
Department of Statistics,
Athens University of Economics and Business
p.dellaportas@ucl.ac.uk

Aristeidis Panos
Department of Statistical Science,
University of College London
aristeidis.panos.15@ucl.ac.uk

Abstract

We provide a linear time inferential framework for Gaussian processes that supports automatic feature extraction through deep neural networks and low-rank kernel approximations. Importantly, we derive approximation guarantees bounding the Kullback–Leibler divergence between the idealized Gaussian process and one resulting from a low-rank approximation to its kernel under two types of approximations, which result in two instantiations of our framework: *Deep Fourier Gaussian Processes*, resulting from random Fourier feature low-rank approximations, and *Deep Mercer Gaussian Processes*, resulting from truncating the Mercer expansion of the kernel. We do extensive experimental evaluation of these two instantiations in a broad collection of real-world datasets providing strong evidence that they outperform a broad range of state-of-the-art methods in terms of time efficiency, negative log-predictive density, and root mean squared error.

1 Introduction

Gaussian Processes (GPs) have long been studied in probability and statistics; e.g. [33]. In Bayesian inference, they provide a canonical way to define a probability distribution over functions, which can be used as a prior to build probabilistic frameworks for quantifying uncertainty in prediction. Among many applications, they have been a method of choice for hyperparameter tuning in deep learning.

In the simplest setting, a zero-mean probability distribution over functions $f : \mathbf{x} \mapsto y$ is defined as follows. For any collection $X = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ of feature vectors, it is assumed that their corresponding responses $\mathbf{y} = (y_1, \ldots, y_N)$ are jointly Gaussian, with zero mean and covariance matrix $K(k_\theta, X) := (k_\theta(\mathbf{x}_i, \mathbf{x}_j))_{ij}$, where $k_\theta(\cdot, \cdot)$ is a positive semidefinite kernel indexed by a parameter vector θ. The usual inferential practice is to assume that we do not observe the Gaussian sample directly but additional noise drawn from a zero-mean isotropic Gaussian distribution is added to it prior to our observation. Bayesian inference then proceeds by focusing on estimation of θ and the noise variance distribution as well as to compute predictive distributions of unobserved responses $\mathbf{y}^*$ corresponding to a collection of new feature vectors X^* of interest. These inference tasks require computing the inverse and determinant of the covariance matrix $K(k_\theta, X)$, which naively costs $O(N^3)$ operations (or more precisely matrix multiplication time), making the inferential framework hard to scale computationally beyond a few thousand observations. A second limitation facing the

Preprint. Under review.

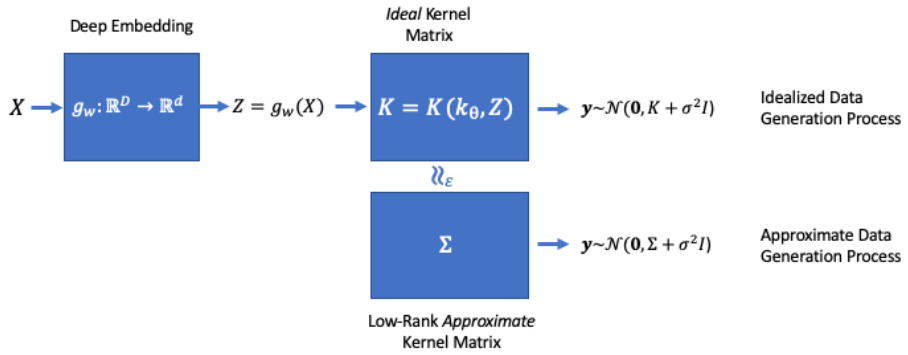

Figure 1: Our (approximate) data generation process. The hyperparameter ε determines the quality of the approximation of the idealized kernel matrix K by the low-rank kernel matrix Σ; in typical applications, ε is dictated by fixing the rank of the approximate kernel matrix. With that fixed, the parameters w, of the DNN, and θ, determining the kernel of the ideal GP, are optimized jointly to maximize the likelihood of $\mathbf{y}$ conditioning on X, under the approximate data generation process.

vanilla application of GPs in Bayesian inference is the necessity to commit a priori to a specific functional form for the kernel, such as the Gaussian and Matérn kernels, and to an a priori chosen distance function $d(\cdot, \cdot)$ measuring the affinity between pairs $\mathbf{x}_i$ and $\mathbf{x}_j$ of feature vectors.

The goal of our framework is two-fold. We want to both improve the running time of GP regression from cubic to linear at a cost of performing approximate rather than exact inference, and to exploit the power of Deep Neural Networks (DNN) in feature discovery. As such, we propose a flexible methodology that simultaneously optimizes with respect to what features to extract and what kernel parameters to use, for a set level of approximation capacity when performing approximate inference. The pieces that compose our overall framework are shown in Figure 1. At a high level, we propose to maximize the likelihood of our data $\mathbf{y}$ conditioning on X, under the assumption that $\mathbf{y} = (y_i)_{i=1}^N$ is sampled from a Gaussian distribution with covariance matrix that can be written as $\Sigma + \sigma^2 I_N$, with Σ being a low-rank matrix that ε-approximates an ideal kernel matrix $K(k_\theta, Z) = (k_\theta(\mathbf{z}_i, \mathbf{z}_j))_{ij}$, computed on embedded feature vectors $Z = (\mathbf{z}_i = g_w(\mathbf{x}_i))_i$ obtained by passing the original feature vectors $\mathbf{x}_i$ through a DNN g_w. A key ingredient in our inference framework is that we estimate jointly the parameters w and θ.

Our main insights are (i) that inference in GPs whose kernel matrix has constant rank can be performed in linear time, avoiding the need for $N \times N$ matrix inversions/determinant computations for the purposes of maximum likelihood estimation/prediction; and (ii) while (i) can be used on its own, whenever the kernel is low-rank, we can also combine (i) with feature extraction using DNNs and low-rank kernel approximation methods into an end-to-end differentiable framework, which similarly does not require $N \times N$ matrix inversion/determinant computation in every step of the back-propagation, or to perform a prediction once the model is trained. The end-to-end differentiability and the light-weight computation required in each back-propagation step, allows for fast training of GP-based models that perform really well compared to state-of-the-art baselines in both prediction power and training/prediction time, as we show in Section 4.

Importantly, we provide theoretical results that provide bounds for the Kullback–Leibler divergence between the idealized and the approximate data generation processes shown in Figure 1. These bounds can become smaller than a desired εN for moderate values of the rank of the approximating kernel Σ.

In the rest of the paper we describe the components of our framework, namely (1) how to conduct GP inference in *linear time*, when the kernel matrix has constant rank (Section 2.1); (2) how to use

DNNs for feature extraction in GPs (Section 2.2); and (3) how to combine (1) and (2) with kernel approximations in Section 2.3. Then in Sections 2.3.2 and 2.3.3 we detail two instantiations of our framework resulting from two different low-rank kernel approximations based on random Fourier features (Section 2.3.2) and truncating the Mercer expansion of the kernel (Section 2.3.3). In both cases, we provide bounds in the KL divergence between the idealized and the approximate data generation process resulting from kernel approximation. In Section 3 we provide a literature review and in Section 4 we test *extensively* the two instantiations of our proposed methodology against many state-of-the-art methods, in a broad collection of real-world datasets.

2 Methodology

2.1 Linear-time inference in Gaussian processes with low-rank kernels

In GP regression, we are given a response vector $\mathbf{y} = (y_i)_{i=1}^N \in \mathbb{R}^N$ whose entries are noisy evaluations of some random function $f(\cdot)$ on a collection of D-dimensional feature vectors $X = (\mathbf{x}_i)_{i=1}^N \in \mathbb{R}^{N\times D}$, i.e. y_i is a noisy observation of $f(\mathbf{x}_i)$.[1] We take the noise, $y_i - f(\mathbf{x}_i)$, for each data entry i to be independent Gaussian with mean 0 and variance σ^2. Moreover, we place a GP prior over $f(\cdot)$, with zero mean and kernel $k_\theta(\cdot,\cdot)$, so that the collection of function values $f(X) := (f(\mathbf{x}_i))_{i=1}^N$ has a joint Gaussian distribution with zero mean and covariance matrix $K(k_\theta, X)$. Setting $A = K(k_\theta, X) + \sigma^2 I_N$, the log-marginal likelihood of the data becomes $\log p(\mathbf{y}|X) = -\frac{1}{2}\mathbf{y}^{\mathrm{T}}A^{-1}\mathbf{y} - \frac{1}{2}\log|A| - \frac{N}{2}\log(2\pi)$. Future observations $\mathbf{y}^* \in \mathbb{R}^{N^*}$ corresponding to feature vectors $X^* = (\mathbf{x}_i^*)_{i=1}^{N^*}$ have a conditional normal distribution with mean and variance given by $\mathbb{E}(\mathbf{y}^*|\mathbf{y}) = K(k_\theta, X^*, X)A^{-1}\mathbf{y}$ and $\mathbf{Var}(\mathbf{y}^*|\mathbf{y}) = K(k_\theta, X^*) + \sigma^2 I_{N^*} - K(k_\theta, X^*, X)A^{-1}K(k_\theta, X^*, X)^{\mathrm{T}}$ respectively, where $K(k_\theta, X^*, X) := (k_\theta(\mathbf{x}_i^*, \mathbf{x}_j))_{ij}$.

Now, suppose that the kernel function $k_\theta(\cdot,\cdot)$ is low-rank in the sense that there exists a feature map $\phi : \mathbb{R}^D \to \mathbb{R}^r$ such that for all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^D$: $k_\theta(\mathbf{x}, \mathbf{x}') = \langle\phi(\mathbf{x}), \phi(\mathbf{x}')\rangle$, where $\langle\cdot,\cdot\rangle$ is the Euclidean inner product. It follows that the kernel matrix $K(k_\theta, X)$ computed on a collection of feature vectors $X = (\mathbf{x}_i)_{i=1}^N$ can be written as $K(k_\theta, X) = \Xi\Xi^{\mathrm{T}}$, where Ξ is an $N \times r$ matrix whose rows are the vectors $\phi(\mathbf{x}_i)$, for $i = 1, \dots N$. As such, we get that the covariance matrix of the data $\mathbf{y}$ is $A = \Xi\Xi^{\mathrm{T}} + \sigma^2 I_N$. We can then use the Woodbury matrix inversion lemma and the Sylvester determinant theorem to obtain explicit forms for the inverse of A and its determinant: $A^{-1} = \sigma^{-2}I_N - \sigma^{-2}\Xi(\sigma^2 I_r + \Xi^{\mathrm{T}}\Xi)^{-1}\Xi^{\mathrm{T}}$, and $|A| = \sigma^{2(N-r)}|\sigma^2 I_r + \Xi^{\mathrm{T}}\Xi|$. Since these identities involve inversion or determinant calculations of $r \times r$ matrices, by plugging them into the expressions for the log-marginal likelihood of observations $\mathbf{y}$ and the mean and variance of the predictive density of future observations $\mathbf{y}^*$, we can, with the right ordering of operations, compute the log-likelihood and the predictive density in $O(r^3 + r^2 N)$ time, i.e. linear in N, when r is a constant.

2.2 Using DNNs for feature extraction in Gaussian processes

We formalize the top 'row' of Fig. 1 by describing how to use DNNs for feature extraction in GPs. We define a distribution over functions mapping feature vectors $\mathbf{x} \in \mathbb{R}^D$ to responses y, which results from the composition of a random function with a deterministic function, as follows. First, a deterministic function $g_w : \mathbf{x} \mapsto \mathbf{z}$ embeds a feature vector $\mathbf{x}$ to a feature vector $\mathbf{z} \in \mathbb{R}^d$; we assume that g_w is parametric, e.g. expressible by a DNN. Next, a random function $h : \mathbf{z} \mapsto y$ is sampled from a GP with noisy observations, as described in Section 2.1, first paragraph. In particular, $f(\cdot)$ is sampled from a GP with mean zero and kernel function $k_\theta : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, and then $y \sim \mathcal{N}(f(\mathbf{z}), \sigma^2)$. Thus, a collection $X = (\mathbf{x}_i)_{i=1}^N$ of feature vectors maps to a collection of responses $\mathbf{y} = (y_i)_{i=1}^N$ sampled as written in the top row of Fig. 1, labeled 'Idealized Data Generation Process,' namely

$$\mathbf{y} \sim \mathcal{N}(0, K(k_\theta, Z) + \sigma^2 I_N), \tag{1}$$

where $Z = (g_w(\mathbf{x}_i) \equiv \mathbf{z}_i)_{i=1}^N$.

[1] We commonly view a collection $X = (\mathbf{x}_i)_{i=1}^N$ of vectors as a matrix whose *rows* are the $\mathbf{x}_i$'s. Similarly a collection $\mathbf{y} = (y_i)_{i=1}^N$ of scalars is viewed as a *column* vector.

2.3 Combining low-rank kernel approximation with deep feature extraction

Consider the setting described in Section 2.2 for some kernel function $k_\theta : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. Clearly, we may take the neural network to be trivial (i.e. the identity function), so whatever we talk about applies equally well to the standard GP regression setting of Section 2.1, first paragraph.

A well-studied topic in mathematics, statistics, and machine learning is approximating kernels with low-rank kernels. Given a kernel function k_θ, the goal is to identify a feature map $\phi_{\theta,\varepsilon} : \mathbb{R}^d \to \mathbb{R}^r$, providing a guarantee of the following form for a collection $Z = (\mathbf{z}_i)_{i=1}^N$ of features vectors in $\mathbb{R}^d$:

$$K(k_\theta, Z) \approx_\varepsilon \Sigma(\phi_{\theta,\varepsilon}, Z), \tag{2}$$

where $\Sigma(\phi_{\theta,\varepsilon}, Z) = (\phi_{\theta,\varepsilon}(\mathbf{z}_i)^{\mathrm{T}}\phi_{\theta,\varepsilon}(\mathbf{z}_j))_{ij}$. Equivalently $\Sigma(\phi_{\theta,\varepsilon}, Z)$ can be written as $\Sigma = \Xi\Xi^{\mathrm{T}}$, where Ξ is a $N \times r$ matrix whose rows are $\phi_{\theta,\varepsilon}(\mathbf{z}_i)$ for $i = 1, \ldots, N$. In particular, Σ is a rank r matrix. Coming back to the setting of Section 2.2/Eq. (1), a collection $X = (\mathbf{x}_i)_{i=1}^N$ of feature vectors maps to a collection of responses $\mathbf{y} = (y_i)_{i=1}^N$, which are *approximately sampled* as follows:

$$\mathbf{y} \sim \mathcal{N}(0, \Sigma + \sigma^2 I_N), \tag{3}$$

as written in the bottom row of Figure 1, labeled 'Approximate Data Generation Process.' In the next few sections we show that (3) can be made to approximate (1) in a precise sense.

2.3.1 Approximation guarantees

We were intentionally vague about what approximation guarantee is pursued in (2), whether this is intended for any set Z of feature vectors or with high probability over sets Z sampled from some measure, and whether there is randomization in the construction of the feature map, the approximation guarantee holding with high probability with respect to this randomization. The reason we were intentionally vague is that there are many results of this form obtaining different guarantees. We discuss two such results in Sections 2.3.2 and 2.3.3, giving rise to two instantiations of our method. Importantly, we show the following general tool which will allow us to bound the KL divergence between (1) and (3), whenever we instantiate the latter with some low-rank kernel Σ.

Theorem 1 (Proof in the Appendix)**.** *Suppose that Σ_1 and Σ_2 are $N \times N$ positive definite (symmetric) matrices, such that $(1+\gamma)\Sigma_1 - \Sigma_2$ is positive semi-definite for some $\gamma \geq 0$. Then*

$$\mathrm{KL}\left[\mathcal{N}(0, \Sigma_1) \,||\, \mathcal{N}(0, \Sigma_2)\right] \leq \frac{1}{2}\mathrm{Tr}(\Sigma_2^{-1/2}(\Sigma_1 - (1-\gamma)\Sigma_2)\Sigma_2^{-1/2}). \tag{4}$$

If additionally $\Sigma_2 \succeq (1+\gamma)^{-1}\Sigma_1$, then we obtain

$$\mathrm{KL}\left[\mathcal{N}(0, \Sigma_1) \,||\, \mathcal{N}(0, \Sigma_2)\right] \leq \gamma N. \tag{5}$$

If $\Sigma_1 = \sigma^2 I_N + K_1$ and $\Sigma_2 = \sigma^2 I_N + K_2$, where K_1 and K_2 are positive semi-definite, $\sigma^2 > 0$, and $(1+\gamma)\Sigma_1 - \Sigma_2$ is positive semi-definite, then

$$\mathrm{KL}\left[\mathcal{N}(0, \Sigma_1) \,||\, \mathcal{N}(0, \Sigma_2)\right] \leq \frac{1}{2\sigma^2}\mathrm{Tr}(K_1 - (1-\gamma)K_2 + \gamma\sigma^2 I_N). \tag{6}$$

Let us instantiate Theorem 1 by taking $K_1 = K(k_\theta, Z)$ and $K_2 = \Sigma(\phi_{\theta,\varepsilon}, Z)$ (which has rank r). K_1 determines the idealized data generation process of (1), while K_2 determines the approximate one of (3). Our theorem states that the KL divergence between these two processes is controlled by (4)–(6), which as we will see in the next sections can become smaller than any desired εN for relatively modest values of the rank r, namely poly-logarithmic in N (Theorem 3), or even an absolute constant (Theorem 5), whenever the dimension d is an absolute constant.

2.3.2 Instantiation No. 1: Deep Fourier Gaussian Processes

A well-studied method for obtaining low-rank kernel approximations is by defining a parametrized family of functions $e_{\boldsymbol{\eta}} : \mathbb{R}^d \to \mathbb{R}$ as well as a distribution $p(\boldsymbol{\eta})$ over $\boldsymbol{\eta}$, defining the feature map $\phi(\mathbf{z}) = (e_{\boldsymbol{\eta}_1}(\mathbf{z}), \ldots, e_{\boldsymbol{\eta}_r}(\mathbf{z}))$ by sampling random $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_r \sim p(\boldsymbol{\eta})$. For example, in a celebrated paper [31], Rahimi and Recht use Bochner's theorem for shift invariant kernels k_θ to define a kernel-specific density $p_\theta(\boldsymbol{\eta})$ such that $e_{\boldsymbol{\eta}}(\cdot)$ is a cosine function with frequency and phase

determined by $\boldsymbol{\eta} \sim p_\theta(\boldsymbol{\eta})$ (derived from a random Fourier feature with spectral frequency $\boldsymbol{\eta}$; see also [6]). The guarantees obtained by [31] for random Fourier features bound the point-wise distance between $k_\theta(\mathbf{z}_i, \mathbf{z}_j)$ and $\phi(\mathbf{z}_i)^{\mathrm{T}}\phi(\mathbf{z}_j)$, for arbitrary $\mathbf{z}_i, \mathbf{z}_j$. To be able to bound the KL divergence between (1) and (3) we need a spectral, rather than an entry-wise, approximation of $\sigma^2 I + K(k_\theta, Z)$ by $\sigma^2 I + \Sigma(\phi, Z)$. These types of results can be obtained as well, as exemplified by the following:

Theorem 2 (Theorem 12 of [2])**.** *Consider the d-dimensional Gaussian kernel $k(\mathbf{z}, \mathbf{z}') = \exp(-2\pi^2||\mathbf{z}-\mathbf{z}'||_2^2)$, and the kernel matrix $K = K(k, Z) = (k(\mathbf{z}_i, \mathbf{z}_j))_{ij}$, where $Z = (\mathbf{z}_1, \ldots, \mathbf{z}_N)$ is a collection of points in $\mathbb{R}^d$ such that, for some $R > 0$, $||\mathbf{z}_i - \mathbf{z}_j||_\infty \leq R, \forall i, j$. Suppose $d \leq 5\log(N/\sigma^2) + 1$ and $\varepsilon \in (0, 1)$. There exists (a samplable in $O(d)$ time) distribution $p(\boldsymbol{\eta})$ and a parametrized family $e_{\boldsymbol{\eta}}(\cdot)$ of* modified Fourier Features *such that, if $r \geq \Omega(\frac{R^d}{\varepsilon^2}(\log\frac{N}{\sigma^2})^{2d}\log(\frac{s_{\sigma^2}(K)}{\delta}))$, where $s_{\sigma^2}(K) = \mathrm{Tr}((\sigma^2 I + K)^{-1}K)$ and $\delta \in (0, 1)$, then the feature map $\phi(\mathbf{z}) = (e_{\boldsymbol{\eta}_1}(\mathbf{z}), \ldots, e_{\boldsymbol{\eta}_r}(\mathbf{z}))$ where $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_r \sim p(\boldsymbol{\eta})$ satisfies the following with probability at least $1 - \delta$:*

$$(1-\varepsilon)(\sigma^2 I_N + K) \preceq (\sigma^2 I_N + \Sigma) \preceq (1+\varepsilon)(\sigma^2 I_N + K), \tag{7}$$

where $\Sigma = (\phi(\mathbf{z}_i)^{\mathrm{T}}\phi(\mathbf{z}_j))_{ij}$, and $\preceq$ denotes semi-definite domination.

Using Theorems 1 and 2 we get the following theorem. We state it for the Gaussian kernel with the same fixed scaling in every direction for notational simplicity. It extends to the general Gaussian kernel with different scaling per direction in an obvious way (rescaling coordinates).

Theorem 3 (Proof in the Appendix)**.** *Consider the setting of Theorem 2, with the same kernel $k(\cdot, \cdot)$, matrix K, dataset Z, radius R, constraint $d \leq 5\log(N/\sigma^2) + 1$, and the same distribution $p(\boldsymbol{\eta})$ and parametric family $e_{\boldsymbol{\eta}}(\cdot)$ of modified Fourier Features used in that theorem. Take $\varepsilon \in (0, \frac{1}{2}]$. If we take $r \geq \Omega(\frac{R^d}{\varepsilon^2}(\log\frac{N}{\sigma^2})^{2d}\log(\frac{N}{\delta}))$ random $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_r \sim p(\boldsymbol{\eta})$ and define the rank r matrix Σ as in Theorem 2, then with probability at least $1 - \delta$, the KL divergence from distribution* (3) *to distribution* (1) *is at most εN.*

Using Random Fourier Features [31], Modified Random Fourier Features [2], or other random feature-based methods to obtain a low-rank approximation to the kernel $K(k_\theta, Z)$ of (1) and instantiate our framework of Section 2.3 gives rise to our family of *Deep Fourier Gaussian Processes (DFGP)*.

2.3.3 Instantiation No. 2: Deep Mercer Gaussian Processes

In this section, we instantiate our framework from Section 2.3, with an alternative approach for obtaining low-rank approximations to the kernel $K(k_\theta, Z)$, namely truncating the Mercer expansion of the kernel [29]. This gives rise to our family of *Deep Mercer Gaussian Processes (DMGP)*. Suppose that k_θ is a Mercer kernel on some probability space $\mathcal{Z} \subseteq \mathbb{R}^d$ with probability measure μ, which means that $k_\theta(\cdot, \cdot)$ can be written as:

$$k_\theta(\mathbf{z}, \mathbf{z}') = \sum_{t=1}^{\infty} \lambda_t e_t(\mathbf{z}) e_t(\mathbf{z}'), \tag{8}$$

where $(\lambda_t)_{t\in\mathbb{N}}$ is a sequence of summable non-negative, non-increasing numbers, i.e. *eigenvalues*, and $(e_t)_{t\in\mathbb{N}}$ is a family of mutually orthogonal unit-norm functions with respect to the inner product $\langle f, g\rangle = \int_{\mathcal{Z}} f(\mathbf{z})g(\mathbf{z})d\mu(\mathbf{z})$, defined by μ, i.e. *eigenfunctions*. Now suppose that $Z = (\mathbf{z}_i)_{i=1}^N$ is a collection of vectors $\mathbf{z}_i \in \mathcal{Z}$. It follows from Eq. (8) that the kernel matrix $K(k_\theta, Z)$ can be written:

$$K(k_\theta, Z) \equiv \sum_{t=1}^{\infty} \lambda_t \boldsymbol{\omega}_t \boldsymbol{\omega}_t^{\mathrm{T}}, \tag{9}$$

where $\boldsymbol{\omega}_t = (e_t(\mathbf{z}_1), e_t(\mathbf{z}_2), \ldots, e_t(\mathbf{z}_N))$, for all $t \in \mathbb{N}$. Recall that the sequence $(\lambda_t)_t$ is summable so $\lambda_t \to 0$ as $t \to \infty$. The rate of convergence is very fast for many kernels. For example, in Appendix B, we illustrate the Mercer expansion of the multi-dimensional Gaussian kernel, whose eigenvalues converge to 0 exponentially fast. This motivates approximating $K(k_\theta, Z)$ by keeping the first few terms of (9), as motivated by the following theorem of [4].

Theorem 4 (Proof of Theorem 4 in [4])**.** *Let $k(\cdot, \cdot)$ be a Mercer kernel on probability space $(\mathcal{Z}, \mu)$ with $k(\mathbf{z}, \mathbf{z}) \leq B$, for all $\mathbf{z} \in \mathcal{Z}$. Let $Z = (\mathbf{z}_1, \ldots, \mathbf{z}_N)$ comprise samples from μ, let $K = K(k, Z)$*

(which satisfies (9)*), and let* $\Sigma = \sum_{t=1}^{r} \lambda_t \boldsymbol{\omega}_t \boldsymbol{\omega}_t^{\mathrm{T}}$*, for some* $r \in \mathbb{N}$ *(which has rank* r*). With probability at least* $1 - \delta$ *over the samples* Z*:*

$$\mathrm{Tr}(K - \Sigma) \leq N \cdot \left(\Lambda_{>r} + \sqrt{\frac{B\Lambda_{>r}}{N\delta}} \right), \tag{10}$$

where $\Lambda_{>r} = \sum_{t>r} \lambda_t$*.*

Using Theorems 1 and 4 we get the following theorem.

Theorem 5 (Proof in the Appendix)**.** *Consider the setting of Theorem 4. Under event* (10) *which occurs with probability at least* $1 - \delta$*, the KL divergence from distribution* (3) *to distribution* (1) *is at most*

$$\frac{N}{2\sigma^2} \cdot \left(\Lambda_{>r} + \sqrt{\frac{B\Lambda_{>r}}{N\delta}} \right). \tag{11}$$

For example, suppose $k(\mathbf{z}, \mathbf{z}') = \exp(-2\pi^2 ||\mathbf{z} - \mathbf{z}'||_2^2)$ *is the multi-variate Gaussian kernel over* $\mathbb{R}^d$*, endowed with a Gaussian density* $\mu(\mathbf{z}) = (2\pi)^{\frac{d}{2}} \exp(-2\pi^2 ||\mathbf{z}||_2^2)$*. Then choosing* $r = (d \log d + \log \frac{1}{\varepsilon\sigma\delta})^{\Omega(d)}$ *makes* (11) *at most* εN*.*

3 Related work

The computational burden of cubic (or more accurately matrix multiplication) time complexity of GP inference has motivated a voluminous literature on faster approximate methods over the last decades; see [27] for a recent survey. Most of these methods rely on the notion of inducing inputs either on the actual Gaussian process domain [30, 35, 36, 17], or the spectral domain [24, 13, 16]. There is also a plethora of works pursuing kernel matrix approximations, either by using the Nyström method [12, 38, 23, 43, 42, 26, 37, 19, 34, 14], or by approximating the kernel function [31, 32, 25, 3, 20, 15]. More flexible GPs that mimic Bayesian hierarchical formulations have been introduced by [7] and further combined with random Fourier features in [6] to obtain a scalable inferential framework.

The idea of combining neural networks with GPs to extract more meaningful representations from high-dimensional data has been used by [18, 5] but both approaches cannot scale to more than a few thousand training points. The work that is closest to ours is [41] in which scalability issues have been dealt with by exploiting Kronecker/Toeplitz algebra combined with the inducing inputs framework and simultaneous estimation of all parameters. While we propose a low-rank kernel approximation, [41] make use of the KISS-GP framework [39]. We perform extensive evaluations of our method against this and several other state-of-the-art scalable GP methods in the next section.

4 Experiments

All experiments were carried out on a Linux machine with 32 2.20GHz CPU cores and 64GB RAM. The implementation of our code is provided at `https://github.com/aresPanos/dmgp_dfgp_regression`.

4.1 Curve learning via low-rank kernel approximations

Appendix D.2 describes an illustrative small data example in which *random Fourier features GP (FGP)* and *Mercer GP (MGP)*, which can be seen as DFGP of Section 2.3.2 and DMGP of Section 2.3.3 equipped with a trivial neural network $g_w(\mathbf{x}) = \mathbf{x}$, are compared against exact Gaussian process regression. Mercer GP achieves identical results; random Fourier features GP tends to provide better point estimates with tighter posterior regions. Robustness to the number of eigenfunctions/spectral frequencies is also illustrated.

4.2 Real data experiments

We compare the following methods: (i) DMGP of Section 2.3.3 with $d = 1$ and $r = 15$; (ii) DFGP of Section 2.3.2 with $d = 4$, $r = 40$, and random Fourier features; (iii) Stochastic Variational

Inference GP with 250 (*SVIGP*) and 500 (*SVIGP+*) inducing points (code used from GPflow [28]) [17]; (iv) Sparse GP Regression with 250 (*SGPR*) and 500 (*SGPR+*) inducing points (code used from GPflow) [36]; (v) Deep Kernel Learning with 5000 (*DKL*) and 10000 (*DKL+*) inducing points and $d = 1$ since we found that larger values of d did not improve performance (code used from `https://gpytorch.ai`) [41]; (vi) Deep GPs with random Fourier features (*RFEDGP*), see [6], with two hidden layers, three GPs per layer, and spectral frequencies being optimized variationally with fixed randomness—we used 20 Monte Carlo samples throughout training since we found it is much faster and as accurate as the training procedure followed by [6] and 100 Monte Carlo samples for prediction as in [6] (code used from `https://github.com/mauriziofilippone/deep_gp_random_features`). All data have been retrieved from UCI repository [8] or the official site of [33].

DMGP and DFGP require joint estimation of the parameters w and θ through maximization of the log marginal likelihood which is a non-decomposable loss function, see [21], so we used the semi-stochastic asynchronous gradient descent suggested in [1]. More details about the practical implementation of DMGP and DFGP are discussed in Appendix C. We emphasise that for maintaining fairness among comparisons, we kept hyperparameter tuning to the minimum for the DNN-based methods, by using, across all datasets, the same [D – 512 – 256 – 64 – d] architecture with hyperbolic tangent activation functions, while the DNN weights of these methods were initialized by pre-training the DNN as suggested by [41, 40]. We ran all methods for 100 epochs using Adam optimizer [22] and mini-batch optimization with mini-batches of size 1000. All GPs used Gaussian kernels with separate length-scale per dimension. All results have been averaged over five random splits (90% train, 10% test).

Table 1: Negative log-predictive density and training time comparison (standard deviations reported in parentheses) on seven standard benchmark real-world datasets; N, N^* and D represent training data size, test data size, and feature dimension, respectively.

	NEGATIVE LOG-PREDICTIVE DENSITY						
	ELEVATORS	PROTEIN	SARCOS	3DROAD	SONG	BUZZ	ELECTRIC
N	14939	41157	44039	391386	463810	524925	1844352
N^*	1660	4573	4894	43488	51535	58325	204928
D	18	9	21	3	90	77	19
SVIGP	0.444(0.021)	1.041(0.007)	−0.422(0.006)	0.652(0.008)	1.208(0.005)	0.087(0.006)	0.804(0.003)
SVIGP+	0.435(0.018)	0.991(0.006)	−0.479(0.004)	0.541(0.008)	1.205(0.005)	0.078(0.005)	0.769(0.002)
SGPR	0.433(0.017)	0.997(0.007)	−0.370(0.007)	0.799(0.007)	1.202(0.006)	0.216(0.005)	0.871(0.002)
SGPR+	0.420(0.017)	0.944(0.005)	−0.468(0.009)	0.737(0.011)	1.198(0.006)	0.186(0.004)	0.810(0.001)
DKL	0.527(0.011)	0.958(0.020)	0.395(0.040)	0.744(0.129)	1.261(0.057)	0.460(0.003)	0.447(0.013)
DKL+	0.536(0.011)	0.961(0.037)	0.430(0.034)	0.687(0.047)	1.315(0.158)	0.438(0.017)	0.448(0.012)
RFEDGP	0.434(0.021)	1.028(0.006)	−0.303(0.061)	0.583(0.009)	1.207(0.006)	0.238(0.032)	0.616(0.004)
DMGP	0.371(0.036)	0.857(0.015)	**−0.777**(0.015)	0.140(0.010)	**1.185**(0.004)	−0.008(0.022)	0.078(0.002)
DFGP	**0.350**(0.029)	**0.853**(0.018)	**−0.777**(0.020)	**0.139**(0.012)	1.189(0.005)	**−0.016**(0.002)	**0.067**(0.004)
DNN+S	0.402(0.030)	0.904(0.013)	−0.559(0.021)	0.239(0.020)	1.211(0.001)	0.019(0.003)	0.165(0.001)
DNN+M	0.401(0.030)	0.893(0.016)	−0.585(0.029)	0.233(0.020)	1.208(0.001)	0.025(0.016)	0.164(0.001)
DNN+F	0.380(0.022)	0.895(0.022)	−0.628(0.044)	0.237(0.008)	1.210(0.002)	0.012(0.001)	0.155(0.001)
	TRAINING TIME (SECONDS)						
SVIGP	59(2)	182(24)	269(19)	2096(297)	2527(19)	2615(165)	8231(302)
SVIGP+	150(5)	425(3)	455(2)	3895(92)	4845(132)	5715(102)	18878(1513)
SGPR	**49**(1)	156(15)	227(11)	1697(53)	2012(13)	2114(109)	11190(68)
SGPR+	144(3)	381(11)	419(7)	3661(124)	4676(118)	5208(336)	30357(573)
DKL	285(27)	435(4)	455(5)	2531(31)	2916(180)	2854(408)	14455(596)
DKL+	774(80)	1317(227)	740(402)	2377(194)	2885(161)	3182(245)	14833(1608)
RFEDGP	184(9)	559(43)	629(49)	2862(296)	4627(66)	4276(232)	26256(1647)
DMGP	121(26)	375(23)	448(26)	3602(238)	3598(95)	3963(63)	14311(134)
DFGP	51(2)	**137**(13)	**146**(1)	**1363**(8)	**1898**(15)	**2092**(26)	**6785**(323)
DNN+S	28(2)	80(7)	78(4)	752(57)	224(8)	513(7)	2211(614)
DNN+M	41(4)	113(10)	113(7)	1061(79)	331(13)	711(12)	3069(841)
DNN+F	53(3)	150(17)	171(13)	1326(93)	245(13)	1890(133)	6504(294)

Table 1 presents comparisons of all methods in terms of NLPD and training time, whereas Appendix D.1 presents comparisons in terms of RMSE, which carry the same message. Both DFGP and DMGP clearly outperform all other methods in speed and NLPD performance. The last three rows of the two sub-tables of Table 1 describe results of extra experiments in which a DNN regression model with RMSE as loss function was first trained on the data, then its fitted outputs Z were independently used as input to fit a Mercer GP (*DNN+M*), random Fourier features GP (*DNN+F*),[2] or simply an isotropic model $\mathbf{y} \sim \mathcal{N}(Z, \sigma^2 I_N)$ (*DNN+S*). These methods do not perform as well in terms of

[2] As discussed in Section 4, Mercer GP and random Fourier features GP are respectively DMGP and DFGP without the neural net.

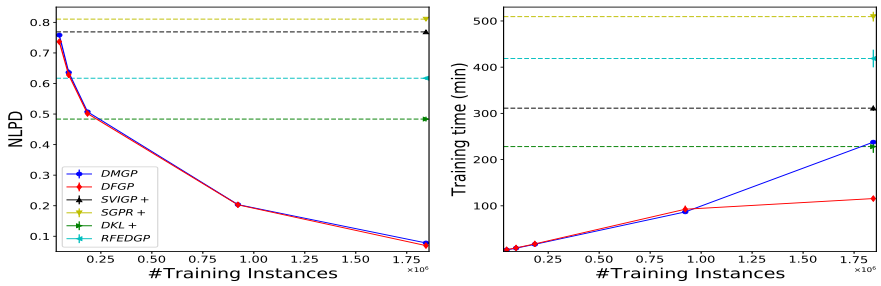

Figure 2: Negative log-predictive density (left) and training times (right) as a function of the number of training points for the ELECTRIC dataset. Dashed lines correspond to baseline models trained on the full dataset and their values can be also found in Table 1.

NLPD, emphasizing the necessity of our suggested joint parameter optimization. However, notice the improvement of the non-parametric *DNN+M* and *DNN+F* over the naive *DNN+S*. We also applied an exact GP regression model (using GPflow) to the smallest dataset ELEVATORS. The average NLPD ($\pm$ one st.d.) was 0.377 ± 0.024 with total average running time 53550 ± 2099 seconds. Comparing with the results of Table 1 we see that both DFGP and NLPD exhibited superior NLPD performance confirming the effectiveness of DNN feature engineering.

Table 2: Comparative negative log-predictive density performance and training time in seconds for different values of rank r and embedding dimension d; standard deviations in parentheses. No results are reported for DMGP for $d = 3$, $\sqrt[3]{r} = 32$ since computational tractability breaks for these values.

	PROTEIN			SARCOS		
	NEGATIVE LOG-PREDICTIVE DENSITY-DMGP					
$\sqrt[d]{r}$	$d=1$	$d=2$	$d=3$	$d=1$	$d=2$	$d=3$
2	0.883(0.014)	0.872(0.012)	0.872(0.015)	−0.778(0.012)	−0.762(0.019)	−0.754(0.029)
4	0.856(0.015)	0.863(0.014)	0.867(0.025)	−0.777(0.015)	−0.775(0.019)	−0.780(0.016)
8	0.857(0.015)	0.855(0.013)	0.848(0.014)	−0.778(0.015)	−0.773(0.021)	−0.780(0.019)
10	0.857(0.015)	0.855(0.013)	0.848(0.014)	−0.777(0.015)	−0.772(0.021)	−0.780(0.020)
16	0.857(0.015)	0.855(0.013)	0.848(0.015)	−0.778(0.015)	−0.772(0.021)	−0.770(0.020)
32	0.857(0.015)	0.855(0.013)	–	−0.777(0.015)	−0.772(0.021)	–
$\frac{r}{2}$	NEGATIVE LOG-PREDICTIVE DENSITY-DFGP					
2	0.871(0.013)	0.873(0.014)	0.862(0.013)	−0.608(0.130)	−0.697(0.069)	−0.771(0.019)
4	0.856(0.013)	0.851(0.012)	0.847(0.014)	−0.784(0.014)	−0.778(0.021)	−0.783(0.019)
8	0.856(0.014)	0.854(0.012)	0.846(0.013)	−0.784(0.014)	−0.779(0.021)	−0.784(0.020)
10	0.856(0.014)	0.855(0.013)	0.846(0.014)	−0.784(0.014)	−0.779(0.020)	−0.786(0.020)
16	0.856(0.014)	0.854(0.012)	0.846(0.015)	−0.784(0.014)	−0.779(0.021)	−0.784(0.022)
32	0.856(0.014)	0.853(0.012)	0.847(0.015)	−0.785(0.014)	−0.781(0.021)	−0.785(0.019)
$\sqrt[d]{r}$	TRAINING TIME-DMGP					
2	115(4)	132(2)	154(2)	127(3)	144(3)	174(3)
4	112(1)	170(3)	374(12)	127(6)	187(2)	417(21)
8	114(1)	308(9)	874(26)	124(6)	325(11)	980(18)
10	116(5)	369(11)	2147(63)	128(5)	401(15)	2325(84)
16	117(1)	404(12)	88649(163)	130(4)	456(16)	94965(293)
32	122(1)	1864(21)	–	135(6)	2071(72)	–
$\frac{r}{2}$	TRAINING TIME-DFGP					
2	108(1)	108(1)	108(1)	124(3)	121(3)	124(3)
4	109(1)	111(2)	110(2)	123(7)	130(2)	126(4)
8	112(1)	112(1)	113(2)	125(2)	132(1)	133(2)
10	115(4)	113(6)	126(3)	127(4)	134(5)	136(3)
16	118(1)	118(2)	156(17)	129(8)	136(4)	188(9)
32	126(1)	126(2)	176(20)	141(5)	145(4)	199(5)

Figure 2 depicts how NLPD and training time over 100 epochs depend on the number of training points in the ELECTRIC dataset, illustrating that our methods can achieve equally good precision with less training points and less time. In particular, notice that DFGP scales better than DMGP.

Table 2 presents the performance of DMGP and DFGP for a series of values of d and r, for the smaller size datasets PROTEIN and SARCOS. Similar results for ELEVATORS dataset can be found in

Appendix D. There is evidence that large values of d and r offer only marginally better performance for both DMGP and DFGP, while severely affecting the training time for DMGP. This suggests using relatively small d and r for DMGP (we used $d = 1, r = 15$ for all our real data experiments) and slightly increase these values for DFGP (we used $d = 4, r = 40$ for all our real data experiments).

4.3 Summary of results

The extensive experiments of this section were designed to answer specific performance questions, the answers to which are summarized here. There is strong evidence that both instantiations of our framework, DFGP and DMGP described in Sections 2.3.2 and 2.3.3 respectively, (i) outperform all state-of-the-art baselines in both time efficiency and prediction accuracy measured in NLPD and RMSE (ii) outperform simple DNN regression without the use of a GP verifying the need for incorporating both our proposed ingredients (iii) achieve competitive performance and are much faster against the competitors with quite fewer training points (iv) outperform exact GP regression inference confirming the importance of the DNN feature extraction (v) illustrate the importance of our proposed joint parameter estimation framework since they clearly outperform consecutive estimation of the DNN first and the kernel parameters after. We also illustrate robustness with respect to r and d and provide practical guidelines.

Broader Impact

Gaussian processes are the principal methodological tool in machine learning that provide probabilistic predictions which are of primary importance in scientific reasoning. Although they have been applied successfully in a series of small data examples in geostatistics, optimisation, data visualisation, robotics, reinforcement learning, spatio-temporal modelling and active learning, their widespread use in big data examples has been hindered due to their poor computational complexity scaling combined with poor feature engineering. Our work a) provides a useful tool for the use of Gaussian processes in big data applications for both academic and applied industry scientists b) does not put anybody in disadvantage c) offers a fail-safe system of big data regression due to automatic feature engineering d) does not leverage any biases in the data.

Acknowledgements

C.D. and P.D. acknowledge partial financial support by the Alan Turing Institute under the EPSRC grant EP/N510129/1. C.D. was supported by NSF Awards IIS-1741137, CCF-1617730 and CCF-1901292, by a Simons Investigator Award, by the DOE PhILMs project (No. DE-AC05-76RL01830), and by the DARPA award HR00111990021. The authors would like to thank Andrew Ilyas for helping them with setting up the Linux machine used for the experiments.

References

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

[2] Haim Avron, Michael Kapralov, Cameron Musco, Christopher Musco, Ameya Velingker, and Amir Zandieh. Random Fourier features for kernel ridge regression: Approximation bounds and statistical guarantees. *arXiv preprint arXiv:1804.09893*, 2018.

[3] Haim Avron, Vikas Sindhwani, Jiyan Yang, and Michael W Mahoney. Quasi-Monte Carlo feature maps for shift-invariant kernels. *The Journal of Machine Learning Research*, 17(1):4096–4133, 2016.

[4] Mikio L Braun. Accurate error bounds for the eigenvalues of the kernel matrix. *The Journal of Machine Learning Research*, 7(11):2303–2328, 2006.

[5] Roberto Calandra, Jan Peters, Carl Edward Rasmussen, and Marc Peter Deisenroth. Manifold Gaussian processes for regression. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 3338–3345. IEEE, 2016.

[6] Kurt Cutajar, Edwin V Bonilla, Pietro Michiardi, and Maurizio Filippone. Random feature expansions for deep Gaussian processes. In *International Conference on Machine Learning*, pages 884–893. JMLR. org, 2017.

[7] Andreas Damianou and Neil Lawrence. Deep Gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215, 2013.

[8] Dheeru Dua and Casey Graff. UCI Machine Learning Repository. `http://archive.ics.uci.edu/ml`, 2017.

[9] Ky Fan. Minimax theorems. *Proceedings of the National Academy of Sciences of the United States of America*, 39(1):42, 1953.

[10] Gregory E Fasshauer. Green's functions: Taking another look at kernel approximation, radial basis functions, and splines. In *Approximation Theory XIII: San Antonio 2010*, pages 37–63. Springer, 2012.

[11] Gregory E Fasshauer and Michael J McCourt. Stable evaluation of Gaussian radial basis function interpolants. *SIAM Journal on Scientific Computing*, 34(2):A737–A762, 2012.

[12] Giancarlo Ferrari-Trecate, Christopher KI Williams, and Manfred Opper. Finite-dimensional approximation of Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 218–224, 1999.

[13] Yarin Gal and Richard Turner. Improving the gaussian process sparse spectrum approximation by representing uncertainty in frequency inputs. In *International Conference on Machine Learning*, pages 655–664, 2015.

[14] Alex Gittens and Michael W Mahoney. Revisiting the Nyström method for improved large-scale machine learning. *The Journal of Machine Learning Research*, 17(1):3977–4041, 2016.

[15] Raffay Hamid, Ying Xiao, Alex Gittens, and Dennis DeCoste. Compact random feature maps. In *International Conference on Machine Learning*, pages 19–27, 2014.

[16] James Hensman, Nicolas Durrande, and Arno Solin. Variational Fourier features for Gaussian processes. *The Journal of Machine Learning Research*, 18(1):5537–5588, 2017.

[17] James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian Processes for Big Data. In *Uncertainty in Artificial Intelligence*, page 282. Citeseer, 2013.

[18] Geoffrey E Hinton and Russ R Salakhutdinov. Using deep belief nets to learn covariance kernels for Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 1249–1256, 2008.

[19] Cho-Jui Hsieh, Si Si, and Inderjit S Dhillon. Fast prediction for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 3689–3697, 2014.

[20] Purushottam Kar and Harish Karnick. Random feature maps for dot product kernels. In *Artificial Intelligence and Statistics*, pages 583–591, 2012.

[21] Purushottam Kar, Harikrishna Narasimhan, and Prateek Jain. Online and stochastic gradient methods for non-decomposable loss functions. In *Advances in Neural Information Processing Systems*, pages 694–702, 2014.

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Ensemble Nyström method. In *Advances in Neural Information Processing Systems*, pages 1060–1068, 2009.

[24] Miguel Lázaro-Gredilla, Joaquin Quiñonero-Candela, Carl Edward Rasmussen, and Aníbal R Figueiras-Vidal. Sparse spectrum Gaussian process regression. *The Journal of Machine Learning Research*, 11:1865–1881, 2010.

[25] Quoc Le, Tamás Sarlós, and Alex Smola. Fastfood-approximating kernel expansions in loglinear time. In *International Conference on Machine Learning*, volume 85, 2013.

[26] Mu Li, James Tin-Yau Kwok, and Baoliang Lü. Making large-scale Nyström approximation possible. In *International Conference on Machine Learning*, page 631, 2010.

[27] Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When Gaussian process meets big data: A review of scalable GPs. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[28] Alexander G. de G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke. Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. GPflow: A Gaussian process library using TensorFlow. *The Journal of Machine Learning Research*, 18(40):1–6, apr 2017.

[29] J Mercer. Functions of positive and negative type and their connection with the theory of integral equations, philosophical transsaction of the royal society of london, ser, 1909.

[30] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, 6(Dec):1939–1959, 2005.

[31] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184, 2008.

[32] Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in Neural Information Processing Systems*, pages 1313–1320, 2009.

[33] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[34] Si Si, Cho-Jui Hsieh, and Inderjit Dhillon. Computationally efficient Nyström approximation using fast transforms. In *International Conference on Machine Learning*, pages 2655–2663, 2016.

[35] Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pages 1257–1264, 2006.

[36] Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574, 2009.

[37] Shusen Wang, Chao Zhang, Hui Qian, and Zhihua Zhang. Improving the modified Nyström method using spectral shifting. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 611–620, 2014.

[38] Christopher KI Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, pages 682–688, 2001.

[39] Andrew Wilson and Hannes Nickisch. Kernel interpolation for scalable structured Gaussian processes (KISS-GP). In *International Conference on Machine Learning*, pages 1775–1784, 2015.

[40] Andrew G Wilson, Zhiting Hu, Russ R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, pages 2586–2594, 2016.

[41] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378, 2016.

[42] Kai Zhang and James T Kwok. Clustered Nyström method for large scale manifold learning and dimension reduction. *IEEE Transactions on Neural Networks*, 21(10):1576–1587, 2010.

[43] Kai Zhang, Ivor W Tsang, and James T Kwok. Improved Nyström low-rank approximation and error analysis. In *International Conference on Machine Learning*, pages 1232–1239, 2008.

[44] Huaiyu Zhu, Christopher KI Williams, Richard Rohwer, and Michal Morciniec. Gaussian regression and optimal finite dimensional linear models. 1997.

Appendix

A Omitted proofs

Proof of Theorem 1: We first show (4). Recall that the KL divergence between two Gaussians with non-singular covariances has a closed form expression:

$$\mathrm{KL}\left[\mathcal{N}(0,\Sigma_1)\,||\,\mathcal{N}(0,\Sigma_2)\right] = \frac{1}{2}\left(\mathrm{Tr}(\Sigma_2^{-1}\Sigma_1) - N + \ln\frac{|\Sigma_2|}{|\Sigma_1|}\right). \tag{12}$$

Because Σ_2 is positive definite, Σ_2^{-1} is too and it has a square root. Thus, by using properties of the trace we can write:

$$\begin{aligned}
\mathrm{Tr}(\Sigma_2^{-1}\Sigma_1) &= \mathrm{Tr}(\Sigma_2^{-1/2}\Sigma_1\Sigma_2^{-1/2}) \\
&= \mathrm{Tr}(\Sigma_2^{-1/2}(\Sigma_1 - (1-\gamma)\Sigma_2 + (1-\gamma)\Sigma_2)\Sigma_2^{-1/2}) \\
&= \mathrm{Tr}(\Sigma_2^{-1/2}(\Sigma_1 - (1-\gamma)\Sigma_2)\Sigma_2^{-1/2}) + \mathrm{Tr}(\Sigma_2^{-1/2}((1-\gamma)\Sigma_2)\Sigma_2^{-1/2}) \\
&= \mathrm{Tr}(\Sigma_2^{-1/2}(\Sigma_1 - (1-\gamma)\Sigma_2)\Sigma_2^{-1/2}) + (1-\gamma)\mathrm{Tr}(I_N) \\
&= \mathrm{Tr}(\Sigma_2^{-1/2}(\Sigma_1 - (1-\gamma)\Sigma_2)\Sigma_2^{-1/2}) + (1-\gamma)N
\end{aligned}$$

Plugging this into (12) yields:

$$\mathrm{KL}\left[\mathcal{N}(0,\Sigma_1)\,||\,\mathcal{N}(0,\Sigma_2)\right] = \frac{1}{2}\left(\mathrm{Tr}(\Sigma_2^{-1/2}(\Sigma_1 - (1-\gamma)\Sigma_2)\Sigma_2^{-1/2}) - \gamma N + \ln\frac{|\Sigma_2|}{|\Sigma_1|}\right). \tag{13}$$

Next we argue the following:

Lemma 6. *If A, B are positive definite, and $B - A$ is positive semidefinite, then* $\ln\left(\frac{|A|}{|B|}\right) \leq 0$.

Proof of Lemma 6: Let $\ell_1 \geq \ell_2 \geq \ldots \geq \ell_N > 0$ be the eigenvalues of A, and $\ell'_1 \geq \ell'_2 \geq \ldots \geq \ell'_N > 0$ be the eigenvalues of B, in non-increasing order. Because $B \succeq A$, by the min-max theorem [9] we have $\ell_i \leq \ell'_i, \forall i$. Thus,

$$\frac{|A|}{|B|} = \prod_{i=1}^{N}\frac{\ell_i}{\ell'_i} \leq 1 \Rightarrow \ln\left(\frac{|A|}{|B|}\right) \leq 0.$$

□

Because $(1+\gamma)\Sigma_1 \succeq \Sigma_2$, it follows from Lemma 6 that

$$0 \geq \ln\left(\frac{|\Sigma_2|}{|(1+\gamma)\Sigma_1|}\right) = \ln\left(\frac{|\Sigma_2|}{(1+\gamma)^N|\Sigma_1|}\right) = \ln\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) - N\ln(1+\gamma) \geq \ln\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) - N\gamma.$$

Combining the last inequality with (13) yields Bound (4).

To prove (5), we note that if additionally $(1+\gamma)\Sigma_2 \succeq \Sigma_1$ then:

$$\frac{1}{2}\mathrm{Tr}(\Sigma_2^{-1/2}((1+\gamma)\Sigma_2 - \Sigma_1)\Sigma_2^{-1/2}) \geq 0. \tag{14}$$

This follows by noticing that matrix $\Sigma_2^{-1/2}((1+\gamma)\Sigma_2 - \Sigma_1)\Sigma_2^{-1/2} \succeq 0$. Indeed, for all $\mathbf{x} \in \mathbb{R}^N$ and using that $(\Sigma_2^{-1/2})^{\mathrm{T}} = \Sigma_2^{-1/2}$:

$$\mathbf{x}^{\mathrm{T}}\Sigma_2^{-1/2}((1+\gamma)\Sigma_2 - \Sigma_1)\Sigma_2^{-1/2}\mathbf{x} = (\Sigma_2^{-1/2}\mathbf{x})^{\mathrm{T}}((1+\gamma)\Sigma_2 - \Sigma_1)(\Sigma_2^{-1/2}\mathbf{x}) \geq 0,$$

where the last inequality follows from the positive semidefiniteness of $(1+\gamma)\Sigma_2 - \Sigma_1$.

Now combining (14) with (4) and using properties of the trace we get:

$$\begin{aligned}
&\mathrm{KL}\left[\mathcal{N}(0,\Sigma_1)\,||\,\mathcal{N}(0,\Sigma_2)\right] \\
&\quad\leq \frac{1}{2}\left(\mathrm{Tr}(\Sigma_2^{-1/2}(\Sigma_1-(1-\gamma)\Sigma_2)\Sigma_2^{-1/2})+\mathrm{Tr}(\Sigma_2^{-1/2}((1+\gamma)\Sigma_2-\Sigma_1)\Sigma_2^{-1/2})\right) \\
&\quad\leq \frac{1}{2}\left(\mathrm{Tr}(\Sigma_2^{-1/2}(2\gamma\Sigma_2)\Sigma_2^{-1/2})\right) \\
&\quad\leq \gamma\mathrm{Tr}(I_N)=\gamma N.
\end{aligned}$$

Let us now move to the proof of (6). We plug $\Sigma_1=\sigma^2 I_N+K_1$ and $\Sigma_2=\sigma^2 I_N+K_2$ into (6) to get:

$$\begin{aligned}
\mathrm{KL}\left[\mathcal{N}(0,\Sigma_1)\,||\,\mathcal{N}(0,\Sigma_2)\right] &\leq \frac{1}{2}\mathrm{Tr}(\Sigma_2^{-1/2}(K_1-(1-\gamma)K_2+\gamma\sigma^2 I_N)\Sigma_2^{-1/2}) \\
&\leq \frac{1}{2}\mathrm{Tr}(\Sigma_2^{-1}(K_1-(1-\gamma)K_2+\gamma\sigma^2 I_N)) \qquad (15)
\end{aligned}$$

where we used properties of the trace. Because K_2 is positive semidefinite, it has eigenvalues $\ell_1\geq\ell_2\geq\ldots\geq\ell_N\geq 0$, which implies that $\Sigma_2=\sigma^2 I+K_2$ has eigenvalues $\sigma^2+\ell_1\geq\sigma^2+\ell_2\geq\ldots\geq\sigma^2+\ell_N>0$, which in turn implies that Σ_2^{-1} has eigenvalues $(\sigma^2+\ell_N)^{-1}\geq(\sigma^2+\ell_{N-1})^{-1}\geq\ldots\geq(\sigma^2+\ell_1)^{-1}>0$. Now using (15) and properties of the trace we have that:

$$\begin{aligned}
\mathrm{KL}\left[\mathcal{N}(0,\Sigma_1)\,||\,\mathcal{N}(0,\Sigma_2)\right] &\leq \frac{1}{2}\mathrm{Tr}(\Sigma_2^{-1}(K_1-(1-\gamma)K_2+\gamma\sigma^2 I_N)) \\
&\leq \frac{1}{2}\lambda_{\max}(\Sigma_2^{-1})\mathrm{Tr}(K_1-(1-\gamma)K_2+\gamma\sigma^2 I_N) \\
&= \frac{1}{2}\cdot\frac{1}{\sigma^2+\ell_N}\cdot\mathrm{Tr}(K_1-(1-\gamma)K_2+\gamma\sigma^2 I_N) \\
&\leq \frac{1}{2\sigma^2}\mathrm{Tr}(K_1-(1-\gamma)K_2+\gamma\sigma^2 I_N),
\end{aligned}$$

where in the above derivation $\lambda_{\max}(\Sigma_2^{-1})$ is the maximum eigenvalue of matrix Σ_2^{-1}. □

Proof of Theorem 3: Set $\Sigma_1=\sigma^2 I_N+K$ and $\Sigma_2=\sigma^2 I_N+\Sigma$. Notice that $s_{\sigma^2}(K)=\mathrm{Tr}((\sigma^2 I+K)^{-1}K)\leq\mathrm{Tr}(I_N)\leq N$. Thus, given our choice of r, Theorem 2 implies that, with probability at least $1-\delta$, Σ_1 and Σ_2 satisfy:

$$(1-\varepsilon)\Sigma_1\preceq\Sigma_2\preceq(1+\varepsilon)\Sigma_1.$$

Given that for $\varepsilon\in(0,\frac{1}{2}]$, we get that $1-\varepsilon\geq\frac{1}{1+2\varepsilon}$, the above implies that:

$$(1+2\varepsilon)^{-1}\Sigma_1\preceq\Sigma_2\preceq(1+2\varepsilon)\Sigma_1.$$

Now we use (5) of Theorem 1, to get that the KL divergence from distribution (3) to distribution (1) is bounded by $2\varepsilon N$. □

Proof of Theorem 5: First, notice that, because Σ is a truncation of K, $K-\Sigma$ is positive semidefinite. To prove (11), we set $K_1=K$, $K_2=\Sigma$, and use (6) from Theorem 1 with $\gamma=0$ to get that the KL divergence from distribution (3) to distribution (1) is bounded by:

$$\frac{1}{2\sigma^2}\mathrm{Tr}(K-\Sigma)\overset{(10)}{\leq}\frac{N}{2\sigma^2}\cdot\left(\Lambda_{>r}+\sqrt{\frac{B\Lambda_{>r}}{N\delta}}\right).$$

To prove the second part of the theorem, we use properties of the spectrum of Gaussian kernels, as discussed in Section B. As per Equations (20), (21), (22), the eigenfunctions and eigenvalues of the Gaussian kernel can be indexed by vectors $\mathbf{n}\in\mathbb{N}^d$. Moreover, the eigenvalues take the form $\lambda_{\mathbf{n}}=c^d\lambda^{\mathbb{1}^{\mathrm{T}}\mathbf{n}}$, for some absolute constants $c>0$ and $\lambda\in(0,1)$, with $\mathbb{1}$ being a vector of all ones. In particular, the eigenvalues are ordered in terms of the "level sets" of $\mathbb{1}^{\mathrm{T}}\mathbf{n}$; namely the larger $\mathbb{1}^{\mathrm{T}}\mathbf{n}$

is, the smaller the eigenvalue is, while every $\mathbf{n}$ with the same value of $\mathbb{1}^{\mathrm{T}}\mathbf{n}$ has the same eigenvalue, $\lambda_{\mathbf{n}} \equiv c^d \lambda^{\mathbb{1}^{\mathrm{T}}\mathbf{n}}$. For $m = \Omega(d \log d + \log \frac{1}{\varepsilon\sigma\delta})$, let us take $r = |\{\mathbf{n} \in \mathbb{N}^d \mid \mathbb{1}^{\mathrm{T}}\mathbf{n} < m\}|$. We have that

$$\begin{aligned}
\Lambda_{>r} &= \sum_{\mathbf{n}:\mathbb{1}^{\mathrm{T}}\mathbf{n} \geq m} \lambda_{\mathbf{n}} \\
&= \sum_{\mathbf{n}:\mathbb{1}^{\mathrm{T}}\mathbf{n} \geq m} c^d \lambda^{\mathbb{1}^{\mathrm{T}}\mathbf{n}} \\
&\leq \sum_{\ell=m}^{\infty} \ell^d c^d \lambda^{\ell} \\
&= c^d \sum_{\ell=m}^{\infty} (\ell^d \lambda^{\ell/2}) \lambda^{\ell/2} \\
&\leq c^d \sum_{\ell=m}^{\infty} \lambda^{\ell/2} \\
&\leq c^d \lambda^{m/2} \cdot \frac{1}{1-\lambda}, \qquad (16)
\end{aligned}$$

where the second to last inequality follows from the fact that $\ell^d \lambda^{\ell/2} \leq 1$ for $m = \Omega(d \log d)$. To conclude the proof notice that the Gaussian kernel $k(\mathbf{z}, \mathbf{z}') = \exp(-2\pi^2 ||\mathbf{z} - \mathbf{z}'||_2^2)$ satisfies $k(\mathbf{z}, \mathbf{z}) = 1$, hence we can use (11) with $B = 1$ to bound the KL divergence from distribution (3) to distribution (1) by

$$\frac{N}{2\sigma^2} \cdot \left(\Lambda_{>r} + \sqrt{\frac{\Lambda_{>r}}{N\delta}} \right) \leq \varepsilon N, \qquad (17)$$

where the last inequality uses (16) and that $m = \Omega(d \log d + \log \frac{1}{\varepsilon\sigma\delta})$. Given that $r = |\{\mathbf{n} \in \mathbb{N}^d \mid \mathbb{1}^{\mathrm{T}}\mathbf{n} < m\}|$, we get that to attain (17) it suffices to choose the rank to be $r = (d \log d + \log \frac{1}{\varepsilon\sigma\delta})^{\Omega(d)}$. □

B Spectrum of the Gaussian kernel

We present the Mercer expansion of the Gaussian kernel:

$$k_{\sigma_f^2,\Delta}(\mathbf{z}_i, \mathbf{z}_j) = \sigma_f^2 \exp(-\frac{1}{2}(\mathbf{z}_i - \mathbf{z}_j)^{\mathrm{T}} \Delta (\mathbf{z}_i - \mathbf{z}_j)), \qquad (18)$$

where $\Delta = \mathrm{diag}(\epsilon_1^2, \ldots, \epsilon_d^2)$ contains the length scales along the d dimensions of the covariates, and σ_f^2 is the variance. In particular, the parameters of the kernel are $\theta = (\sigma_f^2, \Delta)$.

We view $k_{\sigma_f^2,\Delta}(\mathbf{z}_i, \mathbf{z}_j)$ as a kernel over $\mathbb{R}^d$ equipped with an axis aligned Gaussian measure $\rho(\mathbf{z}) = \rho(z^1, \ldots, z^d)$, whose density in dimension j is given by:

$$\rho_j(z^j) = \alpha_j \pi^{-1/2} exp(-\alpha_j^2 (z^j)^2), \ \forall j = 1, \ldots, d. \qquad (19)$$

Mercer's expansion theorem [29] allows us to write

$$k_{\sigma_f^2,\Delta}(\mathbf{z}_i, \mathbf{z}_j) = \sum_{\mathbf{n} \in \mathbb{N}^d} \lambda_{\mathbf{n}} e_{\mathbf{n}}(\mathbf{z}_i) e_{\mathbf{n}}(\mathbf{z}_j), \qquad (20)$$

where $(e_{\mathbf{n}})_{\mathbf{n} \in \mathbb{N}^d}$ is an orthonormal basis of $L_2(\mathbb{R}^d, \rho)$, wherein inner products are computed using $\rho(\mathbf{z})$. It is well-known (see e.g. [44, 33, 10, 11]) that such an orthonormal basis $(e_{\mathbf{n}})_{\mathbf{n} \in \mathbb{N}^d}$ can be constructed as a tensor product of the orthonormal bases of $L_2(\mathbb{R}^d, \rho_j)$ for all j, as follows. Setting

$$\beta_j = \left(1 + (2\epsilon_j/\alpha_j)^2\right)^{1/4}, \ \ \gamma_{n_j} = \sqrt{\frac{\beta_j}{2^{n_j - 1}\Gamma(n_j)}}, \ \ \delta_j^2 = \frac{\alpha_j^2}{2}(\beta_j^2 - 1)$$

the orthonormal eigenvectors are defined as follows

$$e_{\mathbf{n}}(\mathbf{z}) = \prod_{j=1}^{d} e_{n_j}(z^j) = \prod_{j=1}^{d} \left\{ \gamma_{n_j} \exp(-\delta_j^2 (z^j)^2) H_{n_j - 1}(\alpha_j \beta_j z^j) \right\}, \tag{21}$$

where H_n are the Hermite polynomials of degree n; and the corresponding eigenvalues are

$$\lambda_{\mathbf{n}} = \sigma_f^2 \prod_{j=1}^{d} \lambda_{n_j} = \sigma_f^2 \prod_{j=1}^{d} \left\{ \left(\frac{\alpha_j^2}{\alpha_j^2 + \delta_j^2 + \epsilon_j^2} \right)^{1/2} \left(\frac{\epsilon_j^2}{\alpha_j^2 + \delta_j^2 + \epsilon_j^2} \right)^{n_j - 1} \right\}. \tag{22}$$

Note that $\lambda_{n_j} \to 0$ as $n_j \to \infty$. Indeed, as long as $\alpha_j^2 / \epsilon_j^2$ is bounded away from 0, this decay is exponentially fast.

C Implementation details for DMGP and DFGP

We provide further details on how we implement DMGP and DFGP using a Gaussian kernel. In both cases, the crux is to compute the low-rank matrix Σ for a fixed rank r. For DMGP, we compute Σ by using $\sqrt[d]{r} \in \mathbb{N}$ eigenfunctions/eigenvalues per dimension for the Mercer expansion in (20). Therefore, by using the formulas of Section B, we can easily calculate it as

$$\Sigma = \sum_{\mathbf{n} \in \mathbb{N}^d, \mathbf{n} \leq (\sqrt[d]{r}, \ldots, \sqrt[d]{r})} \lambda_{\mathbf{n}} \xi_{\mathbf{n}} \xi_{\mathbf{n}}^\top,$$

where $\xi_{\mathbf{n}} = [e_{\mathbf{n}}(\mathbf{z}_1), \ldots, e_{\mathbf{n}}(\mathbf{z}_N)]^\top \in \mathbb{R}^N$. Note that the parameter a_j in (19) has to be pre-fixed or learnt from the data. We choose to keep it fixed with its value being set $1/\sqrt{2}$ which corresponds to a standard d-dimensional Gaussian measure and we standardize the outputs of DNN, Z, before we feed it as an input to the GP.

Regarding DFGP, we follow the implementation based in algorithm 1 of [31], where we first sample, for even number r, $\frac{r}{2}$ spectral frequencies $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_{\frac{r}{2}}$ from the spectral density $p(\boldsymbol{\eta})$ of the stationary kernel $k_\theta(\cdot, \cdot)$ and then create the feature map $\phi(\mathbf{z}) : \mathbb{R}^d \to \mathbb{R}^r$, defined by

$$\phi(\mathbf{z}) = \sqrt{\frac{2}{r}} \ [\cos(\boldsymbol{\eta}_1^\top \mathbf{z}), \ldots, \cos(\boldsymbol{\eta}_{\frac{r}{2}}^\top \mathbf{z}), \sin(\boldsymbol{\eta}_1^\top \mathbf{z}), \ldots, \sin(\boldsymbol{\eta}_{\frac{r}{2}}^\top \mathbf{z})]^\top.$$

Hence, the rank of Σ is always an even number. The spectral frequencies are only sampled once before training and are then kept fixed throughout optimization of the log-marginal likelihood. Finally, the spectral density in the case of Gaussian kernel in (18) is given by

$$p(\boldsymbol{\eta}) = \frac{\sqrt{|2\pi \Delta^{-1}|}}{\sigma_f^2} \exp(-2\pi^2 \boldsymbol{\eta}^\top \Delta^{-1} \boldsymbol{\eta}).$$

D Additional experimental results

D.1 Extra results on real data

Table 3 demonstrates the RMSE values of all methods. As in Table 1 of the main paper, RMSE values follow similar trends as the corresponding NLPD values, with DMGP and DFGP outperforming all the baselines across all datasets.

Table 4 presents how rank r affects performance and training time over the ELEVATORS dataset where results show similar patterns as in Table 2 of the main paper. Increasing embedding's dimension d does not reduce any further NLPD for both DMGP and DFGP while computational time for DMGP increases fast with d. Similarly, $\sqrt[d]{r}$ and d does not seem to provide any performance boost for values larger than 4 and 2, respectively.

Table 3: RMSE comparison between state-of-the-art baselines and our methods DMGP and DFGP. The experimental set-ups are the same as in Table 1 of the main paper.

	RMSE						
	ELEVATORS	PROTEIN	SARCOS	3DROAD	SONG	BUZZ	ELECTRIC
N	14939	41157	44039	391386	463810	524925	1844352
N^*	1660	4573	4894	43488	51535	58325	204928
D	18	9	21	3	90	77	19
SVIGP	0.379(0.009)	0.683(0.005)	0.160(0.001)	0.462(0.004)	0.810(0.005)	0.271(0.003)	0.540(0.002)
SVIGP+	0.375(0.007)	0.649(0.005)	0.151(0.001)	0.413(0.004)	0.807(0.004)	0.270(0.003)	0.521(0.001)
SGPR	0.375(0.007)	0.653(0.005)	0.168(0.002)	0.537(0.004)	0.806(0.005)	0.315(0.003)	0.577(0.001)
SGPR+	0.370(0.007)	0.620(0.004)	0.153(0.002)	0.506(0.006)	0.802(0.005)	0.308(0.003)	0.542(0.001)
DKL	0.352(0.010)	0.630(0.012)	0.230(0.047)	0.499(0.074)	0.815(0.006)	0.274(0.014)	0.285(0.008)
DKL+	0.361(0.009)	0.632(0.022)	0.276(0.035)	0.474(0.024)	0.813(0.004)	0.268(0.014)	0.296(0.015)
RFEDGP	0.355(0.013)	0.678(0.004)	0.179(0.012)	0.434(0.004)	0.809(0.005)	0.307(0.009)	0.448(0.002)
DMGP	0.346(0.010)	0.564(0.007)	**0.111**(0.002)	**0.277**(0.003)	**0.791**(0.003)	**0.237**(0.000)	0.261(0.001)
DFGP	**0.341**(0.008)	**0.562**(0.008)	**0.111**(0.002)	0.278(0.003)	0.795(0.004)	0.238(0.000)	**0.259**(0.001)
DNN+S	0.359(0.007)	0.588(0.006)	0.144(0.001)	0.311(0.005)	0.806(0.001)	0.251(0.000)	0.288(0.001)
DNN+M	0.359(0.007)	0.581(0.006)	0.140(0.003)	0.310(0.005)	0.804(0.001)	0.250(0.001)	0.287(0.001)
DNN+F	0.354(0.005)	0.582(0.009)	0.135(0.004)	0.311(0.001)	0.805(0.002)	0.250(0.001)	0.285(0.001)

Table 4: Comparative NLPD performance and training time (in seconds) of DMGP and DFGP on ELEVATORS dataset for several values of rank r. No results are reported for DMGP for $d = 3$, $\sqrt[3]{r} = 32$ since computational tractability breaks for these values. Experimental set-ups are the same as in Table 1 of the main paper.

ELEVATORS							
	DMGP				DFGP		
$\sqrt[d]{r}$	$d=1$	$d=2$	$d=3$	$\frac{r}{2}$	$d=1$	$d=2$	$d=3$
			NLPD				
2	0.381(0.037)	0.361(0.032)	0.377(0.044)	2	0.411(0.037)	0.381(0.040)	0.367(0.029)
4	0.371(0.036)	0.351(0.032)	0.353(0.028)	4	0.380(0.037)	0.357(0.032)	0.357(0.030)
8	0.371(0.036)	0.351(0.032)	0.352(0.029)	8	0.379(0.036)	0.356(0.032)	0.356(0.030)
10	0.371(0.037)	0.351(0.032)	0.352(0.029)	10	0.379(0.036)	0.357(0.031)	0.357(0.029)
16	0.371(0.036)	0.351(0.032)	0.352(0.029)	16	0.379(0.037)	0.357(0.032)	0.357(0.030)
32	0.371(0.036)	0.351(0.032)	–	32	0.379(0.036)	0.356(0.032)	0.357(0.030)
			TRAINING TIME				
2	40(1)	49(1)	59(1)	2	39(1)	38(1)	40(0)
4	41(1)	58(1)	135(2)	4	39(1)	38(0)	40(0)
8	41(2)	100(1)	303(5)	8	40(1)	39(1)	41(0)
10	41(2)	117(2)	710(19)	10	41(2)	40(1)	42(1)
16	42(1)	140(3)	31593(151)	16	42(2)	41(0)	51(3)
32	44(2)	620(10)	–	32	46(2)	44(1)	55(3)

D.2 Curve learning via low-rank kernel approximations

We examine the flexibility of our models by comparing them to exact Gaussian process regression models via the following simple example. We generate an artificial dataset based on the function $f(x) = \frac{1}{2}\left(3\sin(2x) + \cos(10x) + \frac{x}{4}\right)$; exact Gaussian process models can easily recover such a smooth function and, therefore, they provide a sound baseline for comparison with our methods. Our simulated dataset has one-dimensional training points $\{x_i, f(x_i)\}_{i=1}^{25}$ where $x_i \sim \mathcal{N}(0, 1)$. We omit to include any DNN for our two methods, i.e. no embedding is being learnt, facilitating thus comparisons with exact Gaussian processes. We call those methods *MGP* and *FGP* since they only depend on Mercer (see Sections B and C from Appendix and 2.3.3 from main paper) and random Fourier features frameworks (see 2.3.2 from main paper), respectively. For all three methods, a Gaussian kernel is used. The exact Gaussian process model has been trained using GPflow.

Figure 3 illustrates how MGP and FGP compare to exact Gaussian process. MGP presents identical behavior, leading to same posterior mean and predictive intervals. The posterior mean of FGP approximates better the underlying curve with more 'confidence' to unseen function values. We use $r = 34$ and $r = 68$ for MGP and FGP respectively.

Figure 4 depicts how MGP and FGP inference is affected by considering different values for r (i.e. eigenfunctions or spectral frequencies) for approximating Σ on the simulated dataset. For MGP, as

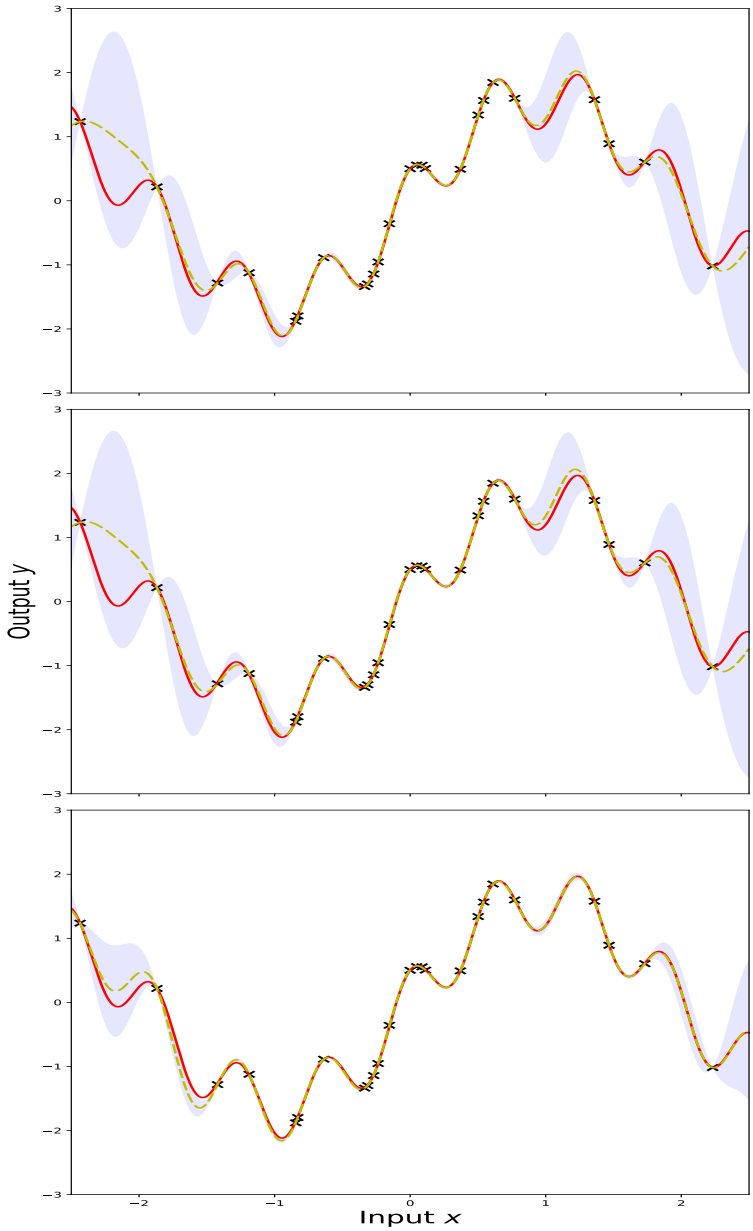

Figure 3: Recovering the function $f(x) = \frac{1}{2}\left(3\sin(2x) + \cos(10x) + \frac{x}{4}\right)$. From top to bottom: Predictive mean and 95% of the predictive probability mass of exact Gaussian process, MGP and FGP, respectively. We make use of 34 eigenfunctions for MGP and 34 spectral frequencies for FGP, i.e. $r = 34$ and $r = 68$, respectively. Black crosses depict the training data, solid red line shows $f(x)$, and the dashed yellow line shows the approximate methods MGP and FGP.

r increases, the uncertainty decreases and posterior mean estimates tend to approximate very well those of the exact GP model. FGP performs well with high confidence even with $r = 4$ and after $r = 24$ learns the true function impressively well.

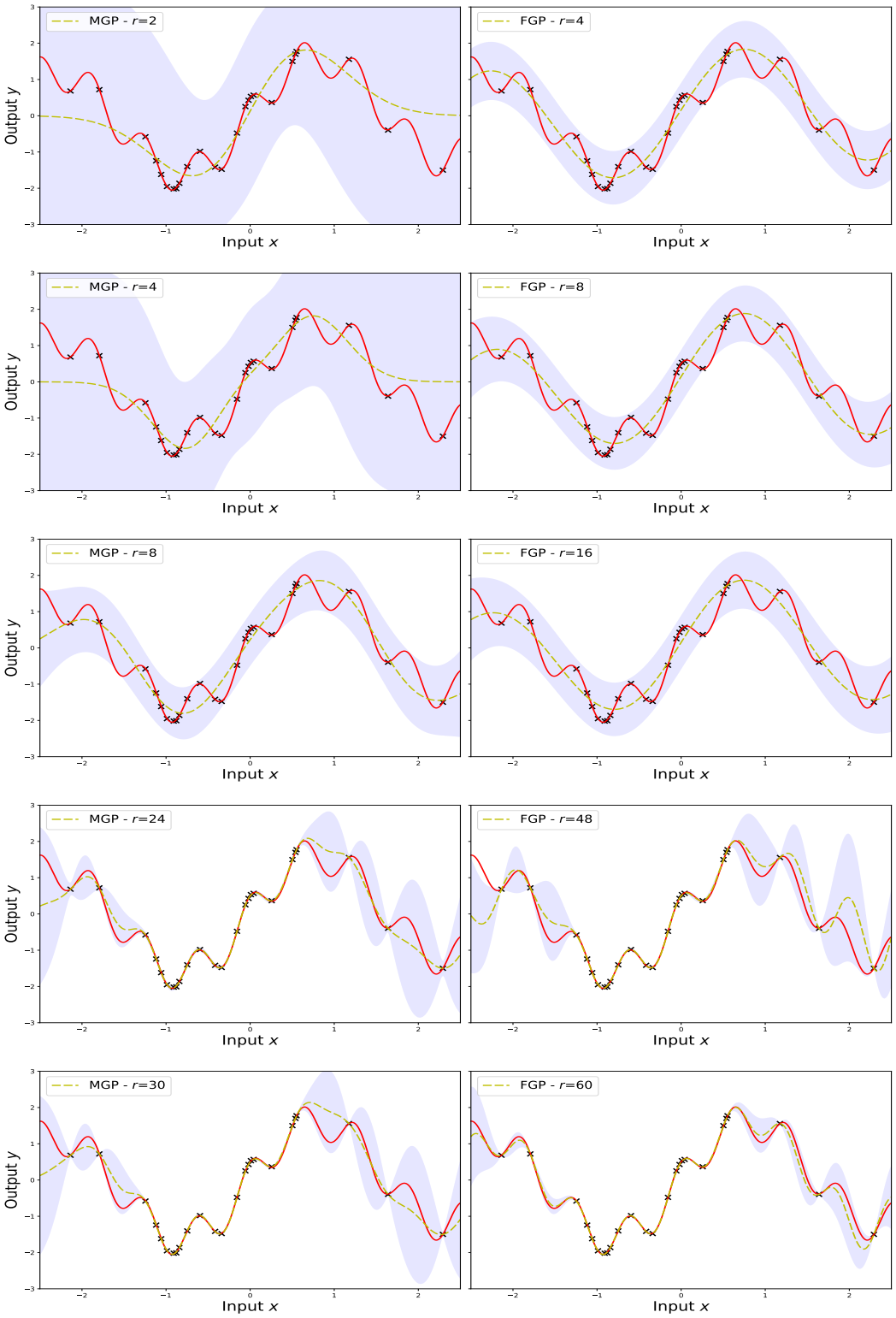

Figure 4: Recovering the function $f(x) = \frac{1}{2}\left(3\sin(2x) + \cos(10x) + \frac{x}{4}\right)$ by using different ranks r for Σ to approximate the true kernel K. Training points are denoted by black crosses, $f(x)$ by solid red line, MGP and FGP with dashed yellow lines.