Scalable Deep Basis Kernel Gaussian Processes

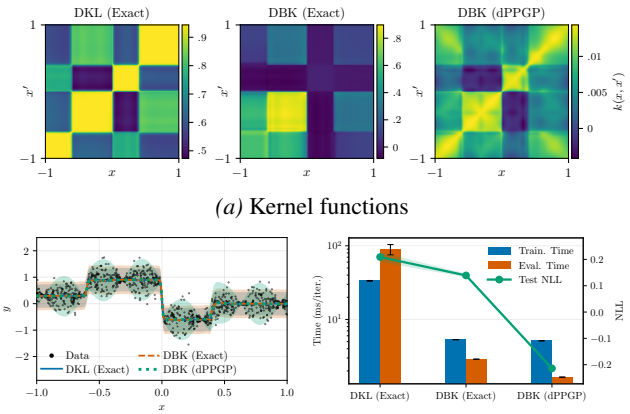
Yunqin Zhu [1] **Henry Shaowu Yuchi** [2] **Yao Xie** [1]

Abstract

Learning expressive kernels while retaining tractable inference remains a central challenge in scaling Gaussian processes (GPs) to large and complex datasets. We propose a scalable GP regressor based on deep basis kernels (DBKs). Our DBK is constructed from a small set of neural-network-parameterized basis functions with an explicit low-rank structure. This formulation immediately enables linear-complexity inference with respect to the number of samples, possibly without inducing points. DBKs provide a unifying perspective that recovers sparse deep kernel learning and Gaussian Bayesian last-layer methods as special cases. We further identify that naively maximizing the marginal likelihood can lead to oversimplified uncertainty and rank-deficient solutions. To address this, we introduce a mini-batch stochastic objective that directly targets the predictive distribution with decoupled regularization. Empirically, DBKs show advantages in predictive accuracy, uncertainty quantification, and computational efficiency across a range of large-scale regression benchmarks.

1. Introduction

Gaussian processes (GPs) provide a principled tool for approximating unknown functions with uncertainty estimates (Rasmussen & Williams, 2005). They have been widely applied in scientific modeling, spatial statistics, experimental design, etc., where uncertainty quantification is critical. Central to the expressiveness of a GP is the choice of its covariance function, or *kernel*, which encodes prior assumptions about the function's structure. While classical kernels such as the radial basis function (RBF) and Matern kernels afford analytical convenience and strong inductive biases, they are often too restrictive to capture the non-stationary, non-monotonic patterns present in complex real-world datasets. Moreover, exact GP inference scales cubically with the number of observations, making it prohibitive for massive datasets. This creates a fundamental tension in GP modeling: learning sufficiently expressive kernels while retaining tractable inference.



(a) Kernel functions

(b) Mean ± 2 Std. *(c)* NLL vs. Time

Figure 1. **Example of fitting a 1-D function.** Data follow a smoothed piecewise-constant mean signal with sinusoidal noise. DBK is the proposed kernel, and dPPGP is the proposed predictive objective for DBK. Training time reports one gradient step (full-batch over 20,000 points for exact GPs; mini-batch of size 200 for dPPGP), and evaluation time reports one batched prediction over 1,000 test points as shown in (b). See Section 4.1 for details.

To overcome the limitations of fixed parametric forms, researchers have pursued more flexible kernels through spectral representations (Wilson & Adams, 2013; Samo & Roberts, 2015; Remes et al., 2017), compositional structures (Duvenaud et al., 2013; Lloyd et al., 2014; Sun et al., 2018), and neural network (NN) parameterizations (Hinton & Salakhutdinov, 2007; Calandra et al., 2016; Wilson et al., 2016a;b; Bradshaw et al., 2017). Among these, deep kernel learning (DKL) (Wilson et al., 2016a) integrates NNs into GPs by encoding the inputs through an NN before applying a base kernel in the latent space. DKL leverages the representational power of NNs to learn data-driven kernels and has shown strong empirical performance.

On the other hand, to reduce the complexity of GP inference, sparse GPs approximate the kernel using a low-rank structure induced by a finite set of inducing points (Williams & Seeger, 2000; Smola & Schökopf, 2000; Lawrence et al.,

[1]School of Industrial and Systems Engineering, Georgia Institute of Technology [2]Los Alamos National Laboratory. Correspondence to: Yao Xie <yao.xie@isye.gatech.edu>.

Accepted at the ICML 2026 Workshop on Connecting Low-rank Representations in AI (CoLoRAI).

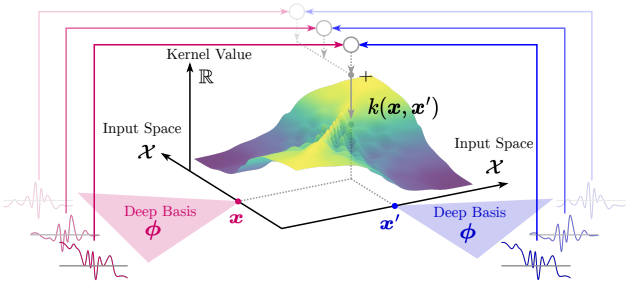


Figure 2. **Illustration of DBK.** A deep architecture $\boldsymbol{\phi}$ maps two inputs $\boldsymbol{x}$ and $\boldsymbol{x}'$ to two sets of basis representations (features) $\{\phi_i(\boldsymbol{x})\}_{i=1}^r$ and $\{\phi_i(\boldsymbol{x}')\}_{i=1}^r$. The kernel value $k(\boldsymbol{x}, \boldsymbol{x}')$ is calculated by summing up the products between two sets of features.

2002; Snelson & Ghahramani, 2005; Quiñonero-Candela & Rasmussen, 2005; Titsias, 2009). This enables scalable inference and mini-batch training, most notably through stochastic variational GPs (SVGPs) (Hensman et al., 2013), which maximize an ELBO of the marginal likelihood. SVGP has been successfully used to deploy sparse DKL (sDKL) on large datasets (Wilson et al., 2016b). However, as noted by Jankowiak et al. (2020), maximum marginal likelihood (MML) training often leads to miscalibrated uncertainty because it mismatches test-time predictions. To address this, a parametric predictive GP (PPGP) objective has been developed that targets the predictive distribution and improves calibration[1] of sDKL.

In this work, we introduce scalable GP regressors based on a family of **D**eep **B**asis **K**ernels (DBKs). DBKs represent the kernel as the inner product of a small set of NN-parameterized basis functions, yielding an explicit low-rank structure. This enables linear-complexity exact inference with respect to the number of training samples. Notably, sparse DKL can be interpreted as a special case of DBKs whose bases are defined implicitly through inducing points and a whitening transformation (Matthews, 2017). DBKs in turn enlarge the design space of deep bases via a two-stage architecture: a neural backbone followed by a flexible expansion layer, possibly without inducing points. This general design also connects to recently developed Gaussian Bayesian last-layer (GBLL) networks (Kristiadi et al., 2020; Watson et al., 2021; Fiedler & Lucia, 2023; Harrison et al., 2024). Specifically, GBLL places a Gaussian prior on the last-layer weights of an NN, equivalent to using the final hidden activations of the network as basis functions. In this sense, DBKs unify sparse sDKL and GBLL within a single low-rank kernel GP framework.

While DBKs provide a flexible and scalable kernel parameterization, the training objective plays a critical role in shaping its uncertainty estimates. We observe that naive MML can lead to degenerate rank-1 solutions that oversimplify predictive uncertainty, especially when the data-generating process involves a heteroskedastic noise. Inspired by PPGP (Jankowiak et al., 2020), we propose dPPGP (where "d" stands for decoupled), a novel mini-batch objective for DBK that directly targets the predictive distribution with decoupled regularization. Specifically, we generalize a common variance correction trick in sparse GPs, which uses an added diagonal term to recover full GP prior variance (Quiñonero-Candela & Rasmussen, 2005; Titsias, 2009). Unlike PPGP, we decouple this correction from test-time prediction and impose it instead as a standalone in-batch trace regularizer during training. This allows flexible control over the regularization strength and makes the objective applicable to arbitrary deep bases. The trace regularization encourages a uniform spread of prior uncertainty, acting as an additional mechanism to prevent rank collapse. Empirically, dPPGP achieves improved uncertainty calibration while retaining the scalability of mini-batch optimization.

Figure 1 compares exact DKL/DBK, and DBK with dPPGP on 1-D synthetic data. While all methods learn expressive kernels, exact GPs fail to encode the sinusoidal noise. Exact DBK collapses towards the rank-1 sample covariance up to a constant mean. In contrast, DBK with dPPGP yields more uniform prior variance, accurately recovers heteroskedastic uncertainty, and achieves the lowest test NLL. The low-rank structure and mini-batch objective also substantially reduce inference latency.

This work makes four contributions. (i) We propose a general low-rank kernel GP regressor with deep bases, unifying sDKL and GBLL models under a single kernel construction. (ii) We theoretically characterize degenerate solutions that arise from MML training of expressive low-rank kernels. (iii) With above insights, we propose a predictive GP objective with decoupled trace regularization, which improves calibration across both inducing-point and non-inducing-point bases. (iv) Experiments on synthetic and real-world regression data show that DBKs are most effective as probabilistic predictors, often improving NLL and CRPS while yielding competitive point-prediction errors.

2. Preliminaries

GPs. A GP defines a prior over functions $f : \mathcal{X} \to \mathbb{R}$ on an input space $\mathcal{X} \subset \mathbb{R}^d$, characterized by a mean function $\mu : \mathcal{X} \to \mathbb{R}$ and a positive semi-definite covariance function, i.e., kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that for any finite set of inputs $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathcal{X}$, the function values are jointly Gaussian distributed as

$$\boldsymbol{f} = [f(\boldsymbol{x}_1), \dots, f(\boldsymbol{x}_n)]^\top \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{X}}, \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}),$$

with $\boldsymbol{\mu}_{\boldsymbol{X}} = [\mu(\boldsymbol{x}_1), \dots, \mu(\boldsymbol{x}_n)]^\top$ and $\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} = [k(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{i,j=1}^n$. We consider $\mu = 0$ w.l.o.g. and observe $\boldsymbol{y} = \boldsymbol{f} + \boldsymbol{\epsilon}$ with i.i.d. noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma_\epsilon^2 \boldsymbol{I}_n)$.

[1] Here, calibration refers to predictive calibration (e.g., correct coverage), not marginal-likelihood optimality.

This gives a marginal $\boldsymbol{y} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{X}})$ with covariance $\boldsymbol{\Sigma}_{\boldsymbol{X}} = \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma_\epsilon^2 \boldsymbol{I}_n$. Exact GP training aims to maximize $\log p(\boldsymbol{y})$. For prediction, a closed-form posterior $f(\boldsymbol{x}^*)|\boldsymbol{y} \sim \mathcal{N}(\hat{\mu}_f(\boldsymbol{x}^*), \hat{\sigma}_f^2(\boldsymbol{x}^*))$ can be derived:

$$\begin{aligned} \hat{\mu}_f(\boldsymbol{x}^*) &= \boldsymbol{k}_{\boldsymbol{X}}(\boldsymbol{x}^*)^\top \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1} \boldsymbol{y}, \\ \hat{\sigma}_f^2(\boldsymbol{x}^*) &= k(\boldsymbol{x}^*, \boldsymbol{x}^*) - \boldsymbol{k}_{\boldsymbol{X}}(\boldsymbol{x}^*)^\top \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1} \boldsymbol{k}_{\boldsymbol{X}}(\boldsymbol{x}^*), \end{aligned} \tag{1}$$

where $\boldsymbol{x}^* \in \mathcal{X}$, $\boldsymbol{k}_{\boldsymbol{X}}(\boldsymbol{x}^*) = [k(\boldsymbol{x}_1, \boldsymbol{x}^*), \ldots, k(\boldsymbol{x}_n, \boldsymbol{x}^*)]^\top$. A major computational bottleneck lies in the inversion of $\boldsymbol{\Sigma}_{\boldsymbol{X}}$, which incurs a cost of $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ space, making standard GP inference impractical for large datasets.

Sparse GPs. The work by Titsias (2009) approximates a full GP with kernel $\tilde{k}$ by a set of r inducing points $\boldsymbol{Z} = \{\boldsymbol{z}_i\}_{i=1}^r \subset \mathcal{X}$ and inducing variables $\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \tilde{\boldsymbol{K}}_{\boldsymbol{Z}\boldsymbol{Z}})$. An approximate low-rank kernel is constructed:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \tilde{\boldsymbol{k}}_{\boldsymbol{Z}}^\top(\boldsymbol{x}) \tilde{\boldsymbol{K}}_{\boldsymbol{Z}\boldsymbol{Z}}^{-1} \tilde{\boldsymbol{k}}_{\boldsymbol{Z}}(\boldsymbol{x}') = \langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x}') \rangle. \tag{2}$$

where $\boldsymbol{\phi}(\boldsymbol{x}) = \tilde{\boldsymbol{K}}_{\boldsymbol{Z}\boldsymbol{Z}}^{-1/2} \tilde{\boldsymbol{k}}_{\boldsymbol{Z}}(\boldsymbol{x})$. This is equivalent to letting $y = \tilde{\boldsymbol{k}}_{\boldsymbol{Z}}^\top(\boldsymbol{x}) \tilde{\boldsymbol{K}}_{\boldsymbol{Z}\boldsymbol{Z}}^{-1} \boldsymbol{u}$. Sparse GP training aims to maximize the lower bound (derived in Appendix F.5):

$$\log \tilde{p}(\boldsymbol{y}) \geq \log p(\boldsymbol{y}) - \sum_{\boldsymbol{x}} \frac{\tilde{k}(\boldsymbol{x}, \boldsymbol{x}) - \|\boldsymbol{\phi}(\boldsymbol{x})\|^2}{2\sigma_\epsilon^2}, \tag{3}$$

where $\tilde{p}(\boldsymbol{y})$ is the marginal likelihood of the full GP defined by $\tilde{k}$. Posterior variance is corrected with a diagonal term:

$$\tilde{\hat{\sigma}}_f^2(\boldsymbol{x}^*) = \hat{\sigma}_f^2(\boldsymbol{x}^*) + \tilde{k}(\boldsymbol{x}^*, \boldsymbol{x}^*) - \|\boldsymbol{\phi}(\boldsymbol{x}^*)\|^2. \tag{4}$$

Evaluation of $\log p(\boldsymbol{y})$, $\nabla \log p(\boldsymbol{y})$, $\hat{\mu}_f(\boldsymbol{x}^*)$, $\tilde{\hat{\sigma}}_f^2(\boldsymbol{x}^*)$ takes $\mathcal{O}(nr^2)$ time and $\mathcal{O}(nr)$ space, using low-rank properties.

SVGP and PPGP. SVGP (Hensman et al., 2013) reduces each training step to $\mathcal{O}(br^2)$ time and $\mathcal{O}(br)$ memory for batch size b. With whitening $\boldsymbol{w} = \tilde{\boldsymbol{K}}_{\boldsymbol{Z}\boldsymbol{Z}}^{-1/2} \boldsymbol{u}$ (Matthews, 2017; Jankowiak et al., 2020) and variational distribution $q(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{m}, \boldsymbol{L}\boldsymbol{L}^\top)$, the ELBO of $\log p(\boldsymbol{y})$ can be written as

$$\begin{aligned} \log p(\boldsymbol{y}) &\geq \mathbb{E}_{q(\boldsymbol{w})}[\log p(\boldsymbol{y}|\boldsymbol{w})] - \mathbb{D}_{\mathrm{KL}}[q(\boldsymbol{w}) \| p(\boldsymbol{w})], \\ &= \sum_{(\boldsymbol{x}, y)} \left\{ \log \mathcal{N}(y; \hat{\mu}_f(\boldsymbol{x}), \sigma_\epsilon^2) - \frac{\hat{\sigma}_f^2(\boldsymbol{x})}{2\sigma_\epsilon^2} \right\} \\ &\quad - \mathbb{D}_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{m}, \boldsymbol{L}\boldsymbol{L}^\top) \| \mathcal{N}(\mathbf{0}, \boldsymbol{I}_r)), \end{aligned}$$

where $\hat{\mu}_f(\boldsymbol{x}) = \langle \boldsymbol{m}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle$, $\hat{\sigma}_f^2(\boldsymbol{x}) = \|\boldsymbol{L}^\top \boldsymbol{\phi}(\boldsymbol{x})\|^2$. Applying (3) and (4), the SVGP loss is given by

$$\begin{aligned} \mathcal{L}_{\text{SVGP}} = \frac{1}{b} \sum_{(\boldsymbol{x}, y)} &\left\{ -\log \mathcal{N}(y; \hat{\mu}_f(\boldsymbol{x}), \sigma_\epsilon^2) + \frac{\tilde{\hat{\sigma}}_f^2(\boldsymbol{x})}{2\sigma_\epsilon^2} \right\} \\ &+ \frac{\beta}{n} \mathbb{D}_{\mathrm{KL}} \left(\mathcal{N}(\boldsymbol{m}, \boldsymbol{L}\boldsymbol{L}^\top) \| \mathcal{N}(0, \boldsymbol{I}_r) \right) \end{aligned} \tag{5}$$

where $\hat{\mu}_f(\boldsymbol{x}) = \langle \boldsymbol{m}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle$ and $\hat{\sigma}_f^2(\boldsymbol{x}) = \|\boldsymbol{L}^\top \boldsymbol{\phi}(\boldsymbol{x})\|^2$. SVGP fits the data using only σ_ϵ^2 in the likelihood term, while the test-time predictive variance contains both $\tilde{\hat{\sigma}}_f^2(\boldsymbol{x})$ and σ_ϵ^2. This mismatch often makes observation noise dominate uncertainty (Jankowiak et al., 2020). PPGP instead directly optimizes the predictive likelihood:

$$\begin{aligned} \mathcal{L}_{\text{PPGP}} = \frac{1}{b} \sum_{(\boldsymbol{x}, y)} &- \log \mathcal{N}(y; \hat{\mu}_f(\boldsymbol{x}), \tilde{\hat{\sigma}}_f^2(\boldsymbol{x}) + \sigma_\epsilon^2) \\ &+ \frac{\beta}{n} \mathbb{D}_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{m}, \boldsymbol{L}\boldsymbol{L}^\top) \| \mathcal{N}(0, \boldsymbol{I}_r)) \end{aligned} \tag{6}$$

where β controls the KL regularization strength. PPGP improves uncertainty quantification but remains tied to inducing-point bases and cannot be applied to a general basis map.

DKL. Deep kernel learning (Wilson et al., 2016a) constructs

$$k(\boldsymbol{x}, \boldsymbol{x}') = \tilde{k}(\boldsymbol{g}(\boldsymbol{x}), \boldsymbol{g}(\boldsymbol{x}')), \tag{7}$$

where $\boldsymbol{g} : \mathcal{X} \to \mathbb{R}^h$ is an NN and $\tilde{k}$ is a base kernel, e.g., RBF or spectral mixture (Wilson & Adams, 2013). DKL leverages learned representations, but the base kernel still imposes a fixed analytic structure. It has been combined with SVGP and PPGP for scalable inference (Wilson et al., 2016b; Jankowiak et al., 2020).

3. Proposed Method

3.1. Deep Basis Kernels

The discussion in Section 2 highlights two recurring themes in modern GP kernel learning: (i) low-rank structure for scalable inference, and (ii) NN parameterization for expressive representation. Deep basis kernels (DBKs) unify these two ideas by constructing kernels directly from a finite set of deep basis functions, yielding an explicit low-rank structure by design. Formally, let $\phi_i : \mathcal{X} \to \mathbb{R}$, $i = 1, \ldots, r$, denote a set of scalar-valued basis functions parameterized by NNs, and define the basis map $\boldsymbol{\phi}(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \ldots, \phi_r(\boldsymbol{x})]^\top$. As in Figure 2, our DBK is given by the inner product:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{i=1}^r \phi_i(\boldsymbol{x}) \phi_i(\boldsymbol{x}') = \langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x}') \rangle, \tag{8}$$

By construction, k is a valid positive semi-definite function (Schölkopf & Smola, 2002). When evaluated at $\boldsymbol{X}$, the kernel matrix can be written compactly as $\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} = \boldsymbol{\Phi}_{\boldsymbol{X}} \boldsymbol{\Phi}_{\boldsymbol{X}}^\top$, where $\boldsymbol{\Phi}_{\boldsymbol{X}} \in \mathbb{R}^{n \times r}$ with rows $\{\boldsymbol{\phi}(\boldsymbol{x}_i)^\top\}_{i=1}^n$, and therefore has rank at most r. The expressiveness of DBKs is grounded in a classical result on kernel representations.

Theorem 3.1 (Mercer). *Any continuous, symmetric, positive semi-definite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ on a compact set*

$\mathcal{X} \subset \mathbb{R}^d$ admits the expansion (Mercer & Forsyth, 1909)

$$k(\boldsymbol{x}, \boldsymbol{x}') = \lim_{r\to\infty} \sum_{i=1}^{r} \lambda_i \psi_i(\boldsymbol{x})\psi_i(\boldsymbol{x}'), \tag{9}$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r \geq 0$ are eigenvalues and $\psi_i : \mathcal{X} \to \mathbb{R}$ are orthonormal eigenfunctions. The series converges absolutely and uniformly on $\mathcal{X} \times \mathcal{X}$.

Theorem 3.1 implies that any kernel can be truncated to a finite-rank expansion. By the universal approximation theorem (Hornik et al., 1989), a deep basis can parameterize $\phi_i(\boldsymbol{x}) = \sqrt{r}\psi_i(\boldsymbol{x})$ arbitrarily well on compact domains. This means that DBKs can in principle approximate any kernel as the rank r and network capacity increase, provided the eigenvalues decay sufficiently fast.

Weight-space view. The kernel (8) has an equivalent weight-space interpretation, which connects low-rank kernel GPs with featurized linear models $f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x})\rangle$, $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_r)$ (Rasmussen & Williams, 2005). In other words, the NN learns a finite feature map, while the GP prior is placed on the coefficients of those features. In the limit of (9), this recovers the Karhunen–Loève expansion of a zero-mean GP (Theorem F.1). This formulation reduces GP inference to Bayesian linear regression in an r-dimensional feature space and immediately allows the closed-form posterior over weights

$$\boldsymbol{w}|\boldsymbol{y} \sim \mathcal{N}\left(\boldsymbol{\Lambda}_{\boldsymbol{X}}^{-1}\boldsymbol{\Phi}_{\boldsymbol{X}}^\top \boldsymbol{y},\ \sigma_\epsilon^2 \boldsymbol{\Lambda}_{\boldsymbol{X}}^{-1}\right), \tag{10}$$

where $\boldsymbol{\Lambda}_{\boldsymbol{X}} = \boldsymbol{\Phi}_{\boldsymbol{X}}^\top \boldsymbol{\Phi}_{\boldsymbol{X}} + \sigma_\epsilon^2 \boldsymbol{I}_r$. The function posterior in (1) hence reduces to $\hat{\mu}_f(\boldsymbol{x}^*) = \boldsymbol{\phi}(\boldsymbol{x}^*)^\top \boldsymbol{\Lambda}_{\boldsymbol{X}}^{-1}\boldsymbol{\Phi}_{\boldsymbol{X}}^\top \boldsymbol{y}$ and $\hat{\sigma}_f^2(\boldsymbol{x}^*) = \sigma_\epsilon^2 \boldsymbol{\phi}(\boldsymbol{x}^*)^\top \boldsymbol{\Lambda}_{\boldsymbol{X}}^{-1}\boldsymbol{\phi}(\boldsymbol{x}^*)$. This can be evaluated in $\mathcal{O}(n(r^2 + c_t))$ time and $\mathcal{O}(n(r + c_s))$ space, where c_t, c_s denote the per-input cost of $\boldsymbol{\phi}$.

MML. DBKs can be trained exactly by maximizing the log marginal likelihood $\log p(\boldsymbol{y})$ in a scalable form:

$$\begin{aligned}\log p(\boldsymbol{y}) = &-\frac{n}{2}\log(2\pi) - \frac{n-r}{2}\log(\sigma_\epsilon^2) - \frac{1}{2}\log|\boldsymbol{\Lambda}_{\boldsymbol{X}}| \\ &- \frac{1}{2\sigma_\epsilon^2}\|\boldsymbol{y}\|_2^2 + \frac{1}{2\sigma_\epsilon^2}\|\boldsymbol{\Lambda}_{\boldsymbol{X}}^{-1/2}\boldsymbol{\Phi}_{\boldsymbol{X}}^\top \boldsymbol{y}\|_2^2,\end{aligned} \tag{11}$$

The full derivation is provided in Appendix F.2. Using the chain rule, the gradient $\nabla \log p(\boldsymbol{y})$ can likewise be computed efficiently. Overall, the low-rank structure reduces the time and space complexity of exact GP inference from $\mathcal{O}(n^3)$ and $\mathcal{O}(n^2)$ to $\mathcal{O}(nr^2)$ and $\mathcal{O}(nr)$, respectively, yielding significant savings when $r \ll n$.

Design of deep bases. A key advantage of DBKs is the flexibility in designing deep basis functions. We adopt a general two-stage architecture that separates representation learning from basis expansion, as illustrated in Figure 3. First, a backbone deep network $\boldsymbol{g} : \mathcal{X} \to \mathbb{R}^h$, e.g., MLP or ResNet, maps the input $\boldsymbol{x}$ to a shared h-dimensional representation $\boldsymbol{g}(\boldsymbol{x})$. Subsequently, a custom expansion layer lifts $\boldsymbol{g}(\boldsymbol{x})$ to r basis functions, where the rank r can be chosen independently of h. Depending on the choice of expansion, this architecture subsumes several closely related methods as special cases:

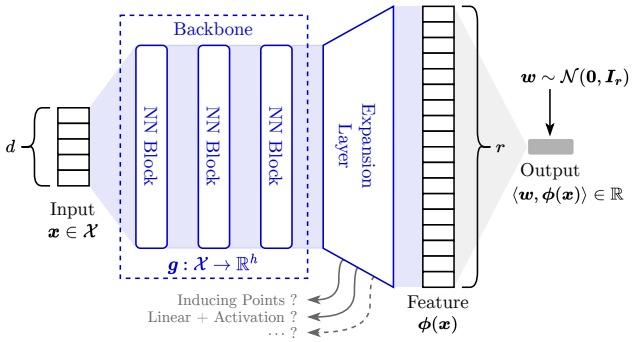


Figure 3. **Network architecture.** A backbone transforms $\boldsymbol{x}$ to an intermediate state, and an expansion layer lifts it to $\boldsymbol{\phi}(\boldsymbol{x})$. Different choices of expansion layers recover sDKL and GBLL.

- **sDKL** (Wilson et al., 2016a;b). Combining the sparse GP approximation in (2) with the deep kernel in (7) yields basis functions of the form

$$\boldsymbol{\phi}(\boldsymbol{x}) = \tilde{\boldsymbol{K}}_{\boldsymbol{Z}\boldsymbol{Z}}^{-1/2}\tilde{\boldsymbol{k}}_{\boldsymbol{Z}}(\boldsymbol{g}(\boldsymbol{x})), \tag{12}$$

  where $\boldsymbol{Z} = \{\boldsymbol{z}_i\}_{i=1}^r \subset \mathbb{R}^h$ are learnable inducing points in the latent space, and $\tilde{k}$ is a base kernel, e.g., RBF or Matérn, with additional parameters, e.g., lengthscales.

- **GBLL** (Kristiadi et al., 2020; Watson et al., 2021; Fiedler & Lucia, 2023; Harrison et al., 2024). These models adapt vanilla NNs by placing a Gaussian prior on the last-layer weights $\boldsymbol{w}$, effectively treating the penultimate-layer activations as $\boldsymbol{\phi}$:

$$\boldsymbol{\phi}(\boldsymbol{x}) = a(\boldsymbol{W}\boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{b}), \tag{13}$$

  where $\boldsymbol{W}$, $\boldsymbol{b}$ are the parameters of a dense linear layer, and $a(\cdot)$ is a standard activation function, e.g. ReLU or SiLU, optionally with element-wise scaling.

Therefore, DBKs give a unifying view of low-rank deep GP kernels. We consider DKL-style inducing-point bases and GBLL-style non-inducing-point bases as two representative cases. Beyond these cases, other expansion layers, such as Fourier features (Rahimi & Recht, 2007; Lázaro-Gredilla et al., 2010) and B-splines (Cunningham et al., 2023), can also be accommodated. We include a small synthetic study of these two additional expansion layers in Appendix E.1. A more systematic study of this broader design space is left to future work.

3.2. dPPGP

The NN parameterization makes DBKs highly flexible by relaxing the analytic structure imposed by standard kernels, but this also makes them more sensitive to the training objective. In particular, we identify that MML can be ill-suited for use with expressive low-rank kernels. To understand this, we analyze the oracle solution of MML under a heteroscedastic regression setting.

Assume fixed inputs $\boldsymbol{X}$ and regression targets $\boldsymbol{y}$ generated from $p_{\text{gt}}(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{\mu}_{\text{gt}}, \boldsymbol{D}_{\text{gt}})$, where $\boldsymbol{\mu}_{\text{gt}} = [\mu_{\text{gt}}(\boldsymbol{x}_1), \dots, \mu_{\text{gt}}(\boldsymbol{x}_n)]^\top$ and $\boldsymbol{D}_{\text{gt}} = \operatorname{diag}(\sigma^2_{\text{gt}}(\boldsymbol{x}_1), \dots, \sigma^2_{\text{gt}}(\boldsymbol{x}_n))$. Here, $\mu_{\text{gt}} : \mathcal{X} \to \mathbb{R}$ and $\sigma^2_{\text{gt}} : \mathcal{X} \to \mathbb{R}^+$ denote the ground-truth mean and variance functions. Under MML, the oracle marginal covariance $\boldsymbol{\Sigma}_{\boldsymbol{X}}$ is characterized by:

Theorem 3.2 (Oracle covariance)**.** *The optimal covariance $\boldsymbol{\Sigma}^*_{\boldsymbol{X}}$ maximizing the expected log marginal likelihood $\mathbb{E}_{p_{\text{gt}}(\boldsymbol{y})}[\log p(\boldsymbol{y})]$ is the sample covariance*

$$\boldsymbol{\Sigma}^*_{\boldsymbol{X}} = \boldsymbol{S}_{\boldsymbol{X}} := \mathbb{E}_{p_{\text{gt}}(\boldsymbol{y})}[\boldsymbol{y}\boldsymbol{y}^\top] = \boldsymbol{\mu}_{\text{gt}}\boldsymbol{\mu}_{\text{gt}}^\top + \boldsymbol{D}_{\text{gt}}. \tag{14}$$

Corollary 3.3 (Rank-1 degeneracy, homogeneous σ^2_{gt})**.** *If $\boldsymbol{D}_{\text{gt}} = \sigma^2_{\text{gt}}\boldsymbol{I}_n$, then under MML, the optimal kernel matrix and noise variance are $\boldsymbol{K}^*_{\boldsymbol{X}\boldsymbol{X}} = \boldsymbol{\mu}_{\text{gt}}\boldsymbol{\mu}_{\text{gt}}^\top$, $\sigma^{*2}_\epsilon = \sigma^2_{\text{gt}}$, respectively. In other words, the kernel solution is rank-1.*

Although a rank-1 solution can perfectly fit the ground-truth mean, it collapses the whole function space onto the same direction. The resulting posterior variance $\hat{\sigma}^2_f$ only measures the signal amplitude and does not capture richer structures for uncertainty. For the general heteroskedastic case, MML leads to a different but still problematic behavior:

Theorem 3.4 (Low-rank solution, heteroskedastic σ^2_{gt})**.** *Let the eigendecomposition of the oracle covariance (14) be $\boldsymbol{S}_{\boldsymbol{X}} = \boldsymbol{U}\operatorname{diag}(\lambda_1, \dots, \lambda_n)\boldsymbol{U}^\top$, $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$. Under MML, the optimal rank-r kernel matrix and noise variance are $\boldsymbol{K}^*_{\boldsymbol{X}\boldsymbol{X}} = \sum_{i=1}^r(\lambda_i - \sigma^{*2}_\epsilon)\boldsymbol{u}_i\boldsymbol{u}_i^\top$ and $\sigma^{*2}_\epsilon = \frac{1}{n-r}\sum_{i=r+1}^n \lambda_i$, respectively, where $\{\lambda_i\}_{i=r+1}^n$ interlace with the diagonal entries of $\boldsymbol{D}_{\text{gt}}$.*

In this solution, the model noise variance σ^2_ϵ can be viewed as a scalar average of the ground-truth σ^2_{gt} across the input space. This means that the model amortizes the heteroskedastic noise via a mis-specified homogeneous component σ^2_ϵ, rather than estimating it with the input-dependent function variance $\hat{\sigma}^2_f$. As a result, the full predictive variance $(\hat{\sigma}^2_f + \sigma^2_\epsilon)$ is largely insensitive to $\boldsymbol{x}$, leading to overconfident predictions in high-noise regions and vice versa.

To address the issues, we generalize ideas from sparse GPs and introduce an improved mini-batch objective, termed dPPGP, where "d" stands for decoupled. Given a variational pseudo-posterior distribution $q(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}; \boldsymbol{m}, \boldsymbol{L}\boldsymbol{L}^\top)$, the predictive mean and function variance of a DBK GP are written as $\hat{\mu}_f(\boldsymbol{x}) = \langle \boldsymbol{m}, \boldsymbol{\phi}(\boldsymbol{x})\rangle$, $\hat{\sigma}^2_f(\boldsymbol{x}) = \|\boldsymbol{L}^\top\boldsymbol{\phi}(\boldsymbol{x})\|^2$. Following PPGP, we define the objective using the log predictive likelihood $\log\mathcal{N}(y; \hat{\mu}_f(\boldsymbol{x}), \hat{\sigma}^2_f(\boldsymbol{x}) + \sigma^2_\epsilon)$ as the data fit term. This formulation forces the full predictive variance to explain input-dependent uncertainty, thereby aligning training more closely with test-time prediction.

A key caveat is that PPGP, as a sparse GP objective, relies on an explicit variance correction derived from the base kernel definition (4), which is not available for a general basis map $\boldsymbol{\phi}$. To this end, we decouple the correction from test time and instead impose it as a regularizer during training:

$$\mathcal{L}_{\text{trace}} = \frac{1}{b}\sum_{\boldsymbol{x}} \frac{\tilde{k}_b - \|\boldsymbol{\phi}(\boldsymbol{x})\|^2}{2\sigma^2_\epsilon},$$
$$\tilde{k}_b = \max_{\boldsymbol{x}} \|\boldsymbol{\phi}(\boldsymbol{x})\|^2$$

where both the sum and maximum are taken over a batch of size b. This regularizer is inherited from (3), with the distinction that the exact kernel diagonal $\tilde{k}(\boldsymbol{x}, \boldsymbol{x})$ is replaced by the in-batch maximum $\tilde{k}_b$, making it applicable to arbitrary deep bases. We refer to this as *trace regularization*, as it is related to the trace of a PSD error matrix (see Appendix F.5). Intuitively, minimizing $\mathcal{L}_{\text{trace}}$ encourages a more uniform allocation of prior variance $\|\boldsymbol{\phi}(\boldsymbol{x})\|^2$ across inputs, providing an additional mechanism to prevent rank collapse.

The complete dPPGP loss is given by

$$\mathcal{L}_{\text{dPPGP}} = \frac{1}{b}\sum_{(\boldsymbol{x},y)} -\log\mathcal{N}(y; \hat{\mu}_f(\boldsymbol{x}), \hat{\sigma}^2_f(\boldsymbol{x}) + \sigma^2_\epsilon) + \alpha\,\mathcal{L}_{\text{trace}} + \frac{\beta}{n}\,\mathbb{D}_{\text{KL}}(\mathcal{N}(\boldsymbol{m}, \boldsymbol{L}\boldsymbol{L}^\top) \,\|\, \mathcal{N}(0, \boldsymbol{I}_r)), \tag{15}$$

where the optimization variables are the parameters of $\boldsymbol{\phi}$, σ^2_ϵ, and the variational parameters $(\boldsymbol{m}, \boldsymbol{L})$; α and β control the strength of the trace regularization and KL regularization, respectively.

4. Experiments

We investigate the empirical performance of DBK GPs across a series of large-scale regression tasks, including a 1-D example, UCI benchmarks, and an application of internet quality estimation. All NN-based models share a ResNet (He et al., 2016; Gorishniy et al., 2021) backbone $\boldsymbol{g}(\cdot)$ with SiLU activation. For standard GPs and DKL, an RBF (base) kernel is used. As per the choice of expansion layers, two DBK variants are considered: (i) **DBK-SiLU**, using a fully connected layer with SiLU activation for basis expansion, i.e., the GBLL case (13); (ii) **DBK-RBF**, using an RBF inducing-point kernel for basis expansion, i.e., the sDKL case (12). A full description of the baselines and detailed settings is available in Appendix E.

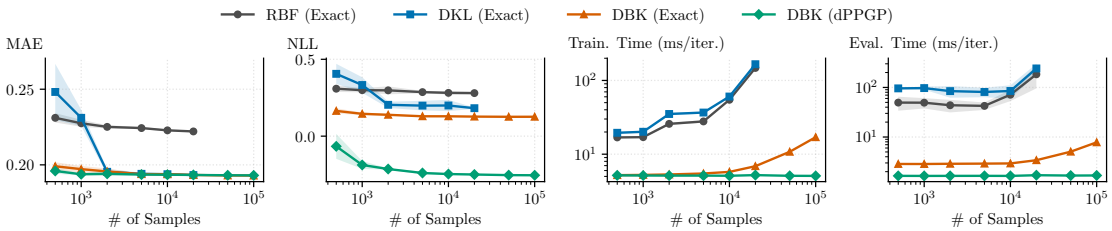


Figure 4. **Scaling behavior on 1-D synthetic data.** We report test MAE and NLL (lower is better). Training time measures a single gradient step (full-batch for exact GPs; mini-batch of size 200 for dPPGP). Evaluation time measures one batched prediction over 1,000 test points. Due to memory constraints, exact RBF and DKL do not scale beyond 20,000 training samples. Shaded regions indicate one standard deviation over 10 random seeds.

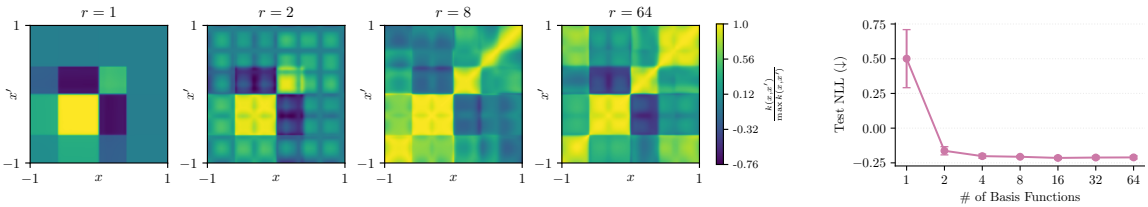


Figure 5. **Effect of rank on 1-D synthetic data.** Left: Learned DBKs of rank $r \in \{1, 2, 8, 64\}$, each normalized by its maximum value on the input grid. Right: Test negative log-likelihood versus rank. Error bars indicate one standard deviation over 10 random seeds. Additional visualizations of the learned basis functions and predictive distributions are provided in Figure 9.

4.1. 1-D Synthetic Data

Setup. We consider a 1-D regression dataset generated by randomly sampling inputs x_i from $\mathcal{U}([-1, 1])$ and targets y_i from $\mathcal{N}(y_i; \mu_{\text{gt}}(x_i), \sigma^2_{\text{gt}}(x_i))$, where $\mu_{\text{gt}}(x)$ is constructed via sharp transitions $\mu_{\text{gt}}(x) = 0.3(1 - s_1) + 0.9(s_1 - s_2) - 0.6(s_2 - s_3)$ with $s_1 = \varsigma(200(x + 0.6))$, $s_2 = \varsigma(200x)$, $s_3 = \varsigma(200(x - 0.4))$, and $\varsigma(\cdot)$ denotes the logistic function. The variance is set to $\sigma^2_{\text{gt}}(x) = (2\sin(10x))^2$, yielding input-dependent uncertainty. We focus on the DBK-SiLU variant in this example, using either exact inference or dPPGP, and compare it against exact RBF and DKL.

Scaling behavior. We vary n from 500 to $100,000$ and report the results in Figure 4. DBK gives the strongest accuracy and calibration, with dPPGP producing a large NLL improvement. These results suggest that non-inducing deep bases can capture sharp, step-like transitions that RBF kernels tend to over-smooth, and that matching the objective to the test-time predictive distribution is important for uncertainty estimation. Exact DBK also reaches $n = 10^5$ with much lower latency due to its linear complexity. dPPGP further improves efficiency through mini-batch variational training, making the approach practical in large-sample regimes. Appendix E.1 compares against a scalable DKL implementation using SKI/KISS-GP (Wilson & Nickisch, 2015).

Effect of rank. We vary the rank r from 1 to 64 and show both the learned kernels and test NLL in Figure 5. Very small ranks are too restrictive: for $r = 1, 2$, the model has high NLL, and the rank-one model learns a single global feature resembling μ_{gt} (Figure 9), producing a blocky kernel that misses local variation. Increasing r makes the kernel more expressive and better captures the sinusoidally varying uncertainty. The gains largely saturate around $r \approx 8$, suggesting that moderate ranks are sufficient for calibrated predictions in this setting.

4.2. UCI Regression

Setup. We evaluate two DBK variants, one with non-inducing bases and one with inducing-point bases, on 10 UCI univariate regression datasets of varying scales[2]. Dataset sizes (n, d) are shown in Table 1. We compare against a plain deep neural network (**DNN**) trained with MSE and three closely related neural GP methods: **VBLL** (Harrison et al., 2024), which maximizes the GBLL ELBO; **SV-DKL** (Wilson et al., 2016b), which trains inducing-point DKL under SVGP; and **PP-DKL** (Jankowiak et al., 2020), which trains inducing-point DKL under PPGP. Both DBK-SiLU and DBK-RBF use the proposed dPPGP objective. These comparisons isolate the role of the predictive objective under closely matched neural low-rank structures. All methods support mini-batch training and, except for DNN, provide sampling-free, single-pass uncertainty estimates.

Benchmark results. Table 1 shows that DBK performs robustly across datasets, with the clearest gains in probabilistic metrics. At least one DBK variant is best or second best on 6 of 10 datasets under MAE, and on all 10 datasets under NLL and CRPS. The controlled comparisons suggest that these gains come primarily from the objective: DBK-

[2] https://archive.ics.uci.edu

Table 1. **UCI regression benchmark.** We report test performance as mean ± standard deviation over 5 random seeds. For each dataset and metric, the best result is shown in **bold** and the second-best is underlined (lower is better). We also report (n, d) for each dataset, where n is the training sample size and d is the input dimension. The last column shows the average rank of each method across all datasets for the corresponding metric. The proposed methods are highlighted in blue.

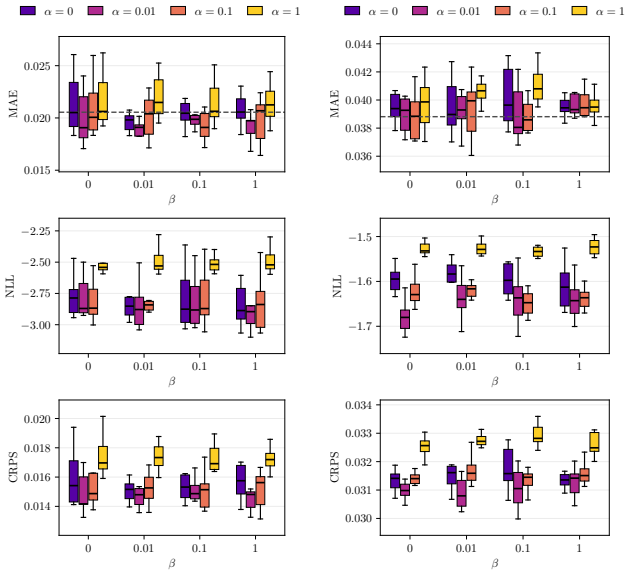
Metric	Method	Pol (12000,26)	Elevators (13279,18)	Kin40K (32000,8)	Protein (36584,9)	Keggdir. (39061,20)	Slice (42800,385)	Keggundir. (50886,27)	3Droad (347898,3)	Song (412275,90)	Buzz (466600,77)	Rank
MAE	DNN	.0222 ± .0009	.2629 ± .0028	**.0250 ± .0004**	.3853 ± .0042	.0392 ± .0015	**.0109 ± .0002**	.0461 ± .0011	.2435 ± .0048	.6072 ± .0007	**.1685 ± .0002**	3.70
	VBLL	.0205 ± .0014	**.2605 ± .0022**	.0268 ± .0013	.3825 ± .0038	.0388 ± .0009	.0133 ± .0028	**.0447 ± .0006**	**.2083 ± .0029**	.6067 ± .0008	.1690 ± .0003	**2.60**
	SV-DKL	.0212 ± .0015	.2636 ± .0049	.0275 ± .0017	**.3703 ± .0041**	.0391 ± .0014	.0139 ± .0023	.0447 ± .0011	.2324 ± .0043	.6062 ± .0010	.1692 ± .0001	3.90
	PP-DKL	.0205 ± .0013	.2614 ± .0020	.0296 ± .0012	.4071 ± .0125	.0392 ± .0018	.0123 ± .0010	.0463 ± .0025	.2527 ± .0037	.6034 ± .0028	.1693 ± .0005	4.70
	DBK-SiLU	.0190 ± .0011	.2606 ± .0025	.0272 ± .0008	.4031 ± .0019	.0386 ± .0018	.0122 ± .0009	.0464 ± .0025	.2285 ± .0088	**.6029 ± .0004**	.1692 ± .0003	3.00
	DBK-RBF	**.0190 ± .0010**	.2614 ± .0024	.0290 ± .0017	.3937 ± .0023	**.0382 ± .0010**	.0123 ± .0020	.0451 ± .0024	.2423 ± .0039	.6036 ± .0015	.1690 ± .0003	3.10
NLL	DNN	–	–	–	–	–	–	–	–	–	–	–
	VBLL	-1.6801 ± .0168	.3623 ± .0080	-1.8664 ± .0420	.8231 ± .0048	-1.0509 ± .0056	-2.1368 ± .3262	-.6915 ± .0020	.2193 ± .0112	1.1732 ± .0016	-.0231 ± .0009	4.30
	SV-DKL	-1.7034 ± .0209	.3607 ± .0115	-1.8244 ± .0370	.8262 ± .0100	-1.0407 ± .0141	-2.2643 ± .3481	-.6870 ± .0025	.3276 ± .0207	1.1737 ± .0012	-.0215 ± .0005	4.70
	PP-DKL	-2.9551 ± .0851	.2989 ± .0054	-2.1023 ± .0231	.7445 ± .0182	-1.6548 ± .0259	-2.4330 ± .1376	-1.7395 ± .0973	-.0801 ± .0151	1.1155 ± .0010	-.1187 ± .0015	2.70
	DBK-SiLU	-2.9670 ± .0550	**.2936 ± .0060**	**-2.1554 ± .0334**	**.7406 ± .0200**	-1.6479 ± .0322	-2.3533 ± .3355	-1.7551 ± .0795	**-.1978 ± .0320**	**1.1126 ± .0010**	-.1181 ± .0020	1.80
	DBK-RBF	**-2.9807 ± .0648**	.2967 ± .0137	-2.1120 ± .0337	.7416 ± .0081	**-1.6679 ± .0423**	**-2.4364 ± .2063**	**-1.7755 ± .1277**	-.1212 ± .0198	1.1136 ± .0013	**-.1202 ± .0014**	**1.50**
CRPS	DNN	–	–	–	–	–	–	–	–	–	–	–
	VBLL	.0215 ± .0005	.1896 ± .0012	.0198 ± .0009	.2884 ± .0023	.0364 ± .0004	.0099 ± .0021	.0508 ± .0002	.1608 ± .0018	.4354 ± .0006	.1265 ± .0001	4.00
	SV-DKL	.0215 ± .0007	.1911 ± .0027	.0206 ± .0010	.2840 ± .0027	.0371 ± .0006	.0103 ± .0017	.0507 ± .0002	.1792 ± .0033	.4354 ± .0006	.1267 ± .0000	4.20
	PP-DKL	.0152 ± .0008	.1873 ± .0009	.0212 ± .0006	.2875 ± .0078	.0317 ± .0005	.0091 ± .0007	.0380 ± .0015	.1766 ± .0025	.4259 ± .0015	.1230 ± .0003	3.30
	DBK-SiLU	.0144 ± .0008	**.1865 ± .0014**	**.0195 ± .0006**	.2852 ± .0015	**.0308 ± .0005**	**.0090 ± .0006**	.0376 ± .0008	**.1599 ± .0059**	**.4253 ± .0002**	.1230 ± .0002	**1.60**
	DBK-RBF	**.0144 ± .0006**	.1870 ± .0014	.0209 ± .0012	**.2817 ± .0019**	.0311 ± .0007	.0091 ± .0015	**.0368 ± .0012**	.1694 ± .0025	.4258 ± .0008	**.1228 ± .0002**	1.90

SiLU differs from VBLL mainly through dPPGP, while DBK-RBF differs from SV-DKL and PP-DKL through the decoupled trace treatment and the same predictive training principle. MAE gains are more modest and can occasionally degrade, consistent with predictive objectives trading some point accuracy for better uncertainty estimates. The variation across datasets also shows that the choice of expansion layer should match the data structure.

Sensitivity analysis. Figure 6 varies the trace weight α and KL weight β for DBK-SiLU on Pol and Keggdir. A small nonzero trace weight improves performance, supporting the role of the decoupled trace regularizer, while the results are less sensitive to β, consistent with Jankowiak et al. (2020). Appendix E.2 gives an additional DBK-RBF analysis and comparison against PP-DKL.

4.3. Mobile Internet Quality Estimation

Setup. Finally, we apply DBK to mobile internet quality estimation in Georgia (GA) and New Mexico (NM) using Ookla speed test data[3] (Jiang et al., 2023). The dataset contains quarterly aggregated upload and download speed measurements from 2019 to 2025, with 517,139 points in GA and 133,512 in NM, rendering exact GPs infeasible. We log-transform targets to stabilize variance, rescale coordinates to $[-1, 1]^2$, and stratify the 10% test set into urban and rural regions using U.S. Census data[4]. We compare DBK-RBF trained with dPPGP against standard RBF and



(a) Pol *(b)* Keggdir.

Figure 6. **Sensitivity to α and β on UCI regression.** We show box plots of the test metrics (lower is better) of DBK-SiLU over 10 random seeds. Dashed lines indicate the VBLL baseline.

DKL baselines trained with SVGP (**SV-RBF/SV-DKL**) to evaluate the benefits of DBK and the predictive objective in a large-scale, real-world application.

Results. Table 2 shows that DBK attains the lowest NLL under all scenarios and improves empirical coverage in rural regions. Figure 8 shows that DBK gives input-dependent 95% PI widths, whereas SV-RBF and SV-DKL produce

[3] https://www.speedtest.net/
[4] https://data.census.gov/

Table 2. **Mobile internet quality estimation.** Mean ± std. over 5 seeds. Lower NLL is better. Cov. is empirical 95% PI coverage.

Dataset	Method	Urban		Rural	
		NLL	Cov.	NLL	Cov.
GA Down.	SV-RBF	1.023 ± .000	96.4 ± .1	1.075 ± .001	94.5 ± .0
	SV-DKL	1.013 ± .002	96.3 ± .0	1.055 ± .002	94.6 ± .0
	DBK	**1.008** ± .001	96.2 ± .1	**1.049** ± .003	94.9 ± .2
GA Up.	SV-RBF	1.025 ± .001	96.1 ± .1	1.195 ± .001	93.4 ± .1
	SV-DKL	1.003 ± .003	95.9 ± .1	1.167 ± .002	93.2 ± .2
	DBK	**.989** ± .003	95.5 ± .2	**1.166** ± .002	94.9 ± .2
NM Down.	SV-RBF	1.016 ± .003	96.4 ± .1	1.138 ± .004	93.6 ± .0
	SV-DKL	1.001 ± .002	96.0 ± .1	1.104 ± .002	93.4 ± .1
	DBK	**.988** ± .002	95.4 ± .1	**1.077** ± .001	94.3 ± .2
NM Up.	SV-RBF	.924 ± .003	96.5 ± .1	1.073 ± .003	93.9 ± .1
	SV-DKL	.893 ± .003	96.0 ± .1	1.011 ± .003	94.1 ± .1
	DBK	**.880** ± .003	95.4 ± .1	**1.005** ± .002	94.9 ± .3

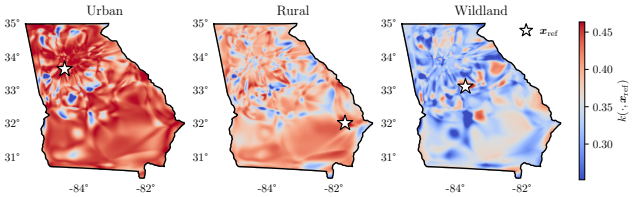
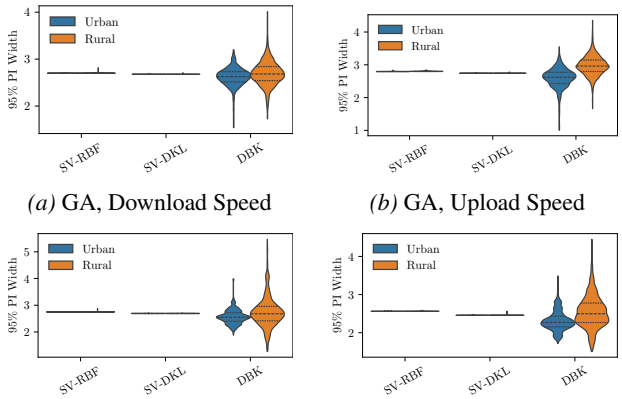


Figure 7. **Kernel slices $k(\cdot, \boldsymbol{x}_{\mathrm{ref}})$ at fixed reference locations.** Results are for DBK on GA download speed, showing non-stationary spatial correlations across urban, rural, and wildland regions.

nearly constant intervals, a typical consequence of overestimated noise σ_ϵ^2. DBK also gives narrower and more concentrated intervals in urban regions than rural regions, reflecting more stable metropolitan network conditions. Figure 7 visualizes DBK kernel slices $k(\cdot, \boldsymbol{x}_{\mathrm{ref}})$ at representative urban, rural, and wildland locations in GA; the induced spatial correlations change markedly with the reference location. These results illustrate that DBK can simultaneously scale to large datasets, express non-stationary spatial correlations, and deliver well-calibrated, interpretable uncertainty estimates in real-world settings.

5. Conclusions and Future Work

We built scalable GPs based on DBKs, a general family of kernels constructed from NN-parameterized basis functions and explicitly enforcing a low-rank structure. We elucidated the design of deep bases and identified the failure modes of standard GP inference under this model class. With these insights, we proposed a mini-batch training objective with decoupled regularization that targets the predictive distribution and prevents rank collapse. Extensive experiments demonstrated the strong performance of DBKs. Appendix A discusses limitations in more detail. An important direction for future work is extending DBKs to non-Gaussian likelihoods (e.g., classification). Furthermore, the rank-1 degeneracy we analyzed arises from treating the regression



(a) GA, Download Speed *(b)* GA, Upload Speed

(c) NM, Download Speed *(d)* NM, Upload Speed

Figure 8. **Violin plots of 95% PI widths.** Model predictions on the test set, split by Urban/Rural. Additional spatial visualizations of predictive uncertainty are provided in Figure 13.

dataset as a single sample path; understanding kernel recovery when multiple sample paths from a ground-truth GP are observed remains an open problem.

References

Achituve, I., Chechik, G., and Fetaya, E. Guided deep kernel learning. In *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, pp. 11–21. PMLR, 2023.

Al-Shedivat, M., Wilson, A. G., Saatchi, Y., Hu, Z., and Xing, E. P. Learning scalable deep kernels with recurrent structure. *Journal of Machine Learning Research*, 18(82): 1–37, 2017. ISSN 1533-7928.

Bradshaw, J., Matthews, A. G. d. G., and Ghahramani, Z. Adversarial examples, uncertainty, and transfer testing robustness in Gaussian process hybrid deep networks, 2017.

Bui, T. D. and Titsias, M. Sparse Gaussian Processes: Structured Approximations and Power-EP Revisited. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Calandra, R., Peters, J., Rasmussen, C. E., and Deisenroth, M. P. Manifold Gaussian processes for regression. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 3338–3345, Vancouver, BC, Canada, 2016. IEEE. ISBN 978-1-5090-0620-5. doi: 10.1109/IJCNN.2016.7727626.

Cunningham, H. J., de Souza, D. A., Takao, S., van der Wilk, M., and Deisenroth, M. P. Actually sparse variational Gaussian processes. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, pp. 10395–10408. PMLR, 2023.

Dai, Y., Zhang, T., Lin, Z., Yin, F., Theodoridis, S., and Cui, S. An interpretable and sample efficient deep kernel for Gaussian process. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 759–768. PMLR, 2020.

Dao, T., De Sa, C. M., and Ré, C. Gaussian quadrature for kernel features. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Dutordoir, V., Durrande, N., and Hensman, J. Sparse Gaussian processes with spherical harmonic features. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 2793–2802. PMLR, 2020.

Duvenaud, D., Lloyd, J., Grosse, R., Tenenbaum, J., and Zoubin, G. Structure discovery in nonparametric regression through compositional kernel search. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 1166–1174. PMLR, 2013.

Fiedler, F. and Lucia, S. Improved uncertainty quantification for neural networks with Bayesian last layer. *IEEE Access*, 11:123149–123160, 2023. ISSN 2169-3536. doi: 10.1109/ACCESS.2023.3329685.

Gardner, J., Pleiss, G., Weinberger, K., Bindel, D., and Wilson, A. G. GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Gorishniy, Y., Rubachev, I., Khrulkov, V., and Babenko, A. Revisiting Deep Learning Models for Tabular Data. In *Advances in Neural Information Processing Systems*, volume 34, pp. 18932–18943. Curran Associates, Inc., 2021.

Harrison, J., Willes, J., and Snoek, J. Variational Bayesian last layers. In *The Twelfth International Conference on Learning Representations*, 2024.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, pp. 282–290, Arlington, Virginia, USA, 2013. AUAI Press.

Hensman, J., Durrande, N., and Solin, A. Variational Fourier features for Gaussian processes. *Journal of Machine Learning Research*, 18(151):1–52, 2018. ISSN 1533-7928.

Hinton, G. E. and Salakhutdinov, R. R. Using deep belief nets to learn covariance kernels for Gaussian processes. In *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

Horn, R. A. and Johnson, C. R. *Matrix analysis*. Cambridge university press, 2012.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90020-8.

Huang, W., Zhao, D., Sun, F., Liu, H., and Chang, E. Scalable Gaussian process regression using deep neural networks. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pp. 3576–3582, Buenos Aires, Argentina, 2015. AAAI Press. ISBN 978-1-57735-738-4.

Jankowiak, M., Pleiss, G., and Gardner, J. Parametric Gaussian Process Regressors. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 4702–4712. PMLR, 2020.

Jiang, H., Yuchi, H. S., Belding, E., Zegura, E., and Xie, Y. Mobile internet quality estimation using self-tuning kernel regression, 2023.

Kristiadi, A., Hein, M., and Hennig, P. Being Bayesian, even just a bit, fixes overconfidence in ReLU networks. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 5436–5446. PMLR, 2020.

Lawrence, N., Seeger, M., and Herbrich, R. Fast sparse Gaussian process methods: The informative vector machine. In *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2002.

Lázaro-Gredilla, M. and Figueiras-Vidal, A. Inter-domain Gaussian processes for sparse inference using inducing features. In *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.

Lazaro-Gredilla, M. and Figueiras-Vidal, A. R. Marginalized neural network mixtures for large-scale regression. *IEEE Transactions on Neural Networks*, 21(8):1345–1351, 2010. ISSN 1941-0093. doi: 10.1109/TNN.2010.2049859.

Lázaro-Gredilla, M., Quiñnero-Candela, J., Rasmussen, C. E., An&#237, and Figueiras-Vidal, b. R. Sparse spectrum Gaussian process regression. *Journal of Machine Learning Research*, 11(63):1865–1881, 2010. ISSN 1533-7928.

Li, K., Balakirsky, M., and Mak, S. Trigonometric quadrature Fourier features for scalable Gaussian process regression. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, pp. 3484–3492. PMLR, 2024.

Liu, J. Z., Padhy, S., Ren, J., Lin, Z., Wen, Y., Jerfel, G., Nado, Z., Snoek, J., Tran, D., and Lakshminarayanan, B. A simple approach to improve single-model deep uncertainty via distance-awareness. *Journal of Machine Learning Research*, 24(42):1–63, 2023. ISSN 1533-7928.

Lloyd, J., Duvenaud, D., Grosse, R., Tenenbaum, J., and Ghahramani, Z. Automatic construction and natural-language description of nonparametric regression models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 28(1), 2014. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v28i1.8904.

Matias, A. L. S., Mattos, C. L. C., Gomes, J. P. P., and Mesquita, D. P. P. Amortized variational deep kernel learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *ICML'24*, pp. 35063–35078, Vienna, Austria, 2024. JMLR.org.

Matthews, A. G. d. G. Scalable Gaussian process inference using variational methods. 2017. doi: 10.17863/CAM.25348.

Mercer, J. and Forsyth, A. R. Functions of positive and negative type, and their connection the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 209(441-458):415–446, 1909. doi: 10.1098/rsta.1909.0016.

Ober, S. W., Rasmussen, C. E., and van der Wilk, M. The promises and pitfalls of deep kernel learning. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, pp. 1206–1216. PMLR, 2021.

Pleiss, G., Gardner, J., Weinberger, K., and Wilson, A. G. Constant-Time Predictive Distributions for Gaussian Processes. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 4114–4123. PMLR, 2018.

Quiñonero-Candela, J. and Rasmussen, C. E. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6(65):1939–1959, 2005. ISSN 1533-7928.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning*. The MIT Press, 2005. ISBN 978-0-262-25683-4. doi: 10.7551/mitpress/3206.001.0001.

Remes, S., Heinonen, M., and Kaski, S. Non-stationary spectral kernels. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Samo, Y.-L. K. and Roberts, S. Generalized spectral kernels, 2015.

Schölkopf, B. and Smola, A. J. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

Smola, A. J. and Schökopf, B. Sparse greedy matrix approximation for machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pp. 911–918, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 978-1-55860-707-1.

Snelson, E. and Ghahramani, Z. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005.

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M. M. A., Prabhat, P., and Adams, R. P. Scalable Bayesian optimization using deep neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 2171–2180, Lille, France, 2015. JMLR.org.

Sun, S., Zhang, G., Wang, C., Zeng, W., Li, J., and Grosse, R. Differentiable compositional kernel learning for Gaussian processes. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 4828–4837. PMLR, 2018.

Tipping, M. E. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1(Jun):211–244, 2001. ISSN ISSN 1533-7928.

Titsias, M. Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, pp. 567–574. PMLR, 2009.

Ton, J.-F., Flaxman, S., Sejdinovic, D., and Bhatt, S. Spatial mapping with Gaussian processes and nonstationary Fourier features. *Spatial Statistics*, 28:59–78, 2018. ISSN 22116753. doi: 10.1016/j.spasta.2018.02.002.

Tran, G.-L., Bonilla, E. V., Cunningham, J., Michiardi, P., and Filippone, M. Calibrating deep convolutional Gaussian processes. In *Proceedings of the Twenty-Second*

International Conference on Artificial Intelligence and Statistics, pp. 1554–1563. PMLR, 2019.

van Amersfoort, J., Smith, L., Jesson, A., Key, O., and Gal, Y. On feature collapse and deep kernel learning for single forward pass uncertainty, 2022.

Wang, K., Wang, G., and Li, D. Scalable Gaussian process via Hilbert-Schmidt singular value decomposition. 2024.

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

Williams, C. and Seeger, M. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000.

Wilson, A. and Adams, R. Gaussian process kernels for pattern discovery and extrapolation. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 1067–1075. PMLR, 2013.

Wilson, A. and Nickisch, H. Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP). In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1775–1784. PMLR, 2015.

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

Wilson, A. G., Hu, Z., Salakhutdinov, R. R., and Xing, E. P. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016b.

Yang, Z., Wilson, A., Smola, A., and Song, L. À la carte – learning fast kernels. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pp. 1098–1106. PMLR, 2015.

Zhao, W., Chen, H., Liu, T., Tuo, R., and Tian, C. From Deep Additive Kernel Learning to Last-Layer Bayesian Neural Networks via Induced Prior Approximation. In *The 28th International Conference on Artificial Intelligence and Statistics*, 2025.

A. Limitations

This work has several limitations.

- The present formulation and experiments focus on Gaussian regression. Extending DBKs and dPPGP to non-Gaussian likelihoods, including classification, requires additional approximation choices and is left for future work.

- Our analysis characterizes objective-induced behavior on a fixed regression dataset, treating the observed labels as one sample from a distribution with fixed mean and noise functions. A hierarchical setting with multiple sample paths from a latent ground-truth GP would lead to a different notion of kernel recovery.

- dPPGP uses a diagonal, training-time trace regularizer rather than a full covariance correction. This preserves the low-rank computational structure and makes the objective applicable to arbitrary deep bases, but it does not cover richer block-diagonal or full-covariance corrections used in some sparse GP approximations (Quiñonero-Candela & Rasmussen, 2005; Bui & Titsias, 2025).

- dPPGP is designed to improve probabilistic prediction and calibration. Objectives that emphasize predictive uncertainty can slightly trade off point-prediction metrics such as MAE, especially when the predictive likelihood can reduce loss by allocating input-dependent variance.

B. Broader Impact

This paper presents methodological work on scalable Gaussian process regression with improved uncertainty quantification. It may benefit scientific modeling, spatial analysis, and large-scale regression tasks where calibrated uncertainty is useful. As with other predictive models, misuse in high-stakes settings without domain validation, or use on biased and unevenly sampled spatial data, could lead to misleading conclusions. The work does not introduce new data, deployment mechanisms, or application-specific decision rules.

C. Related Works

Deep kernel GPs. A long line of works integrate NNs into GP models by learning data-dependent representations or kernels. Early approaches apply standard kernels to learned feature spaces (Hinton & Salakhutdinov, 2007; Calandra et al., 2016). Deep kernel learning (DKL) (Wilson et al., 2016a) formalizes this idea by composing a deep NN map with a base kernel, often combined with expressive spectral kernels (Wilson & Adams, 2013) and scalable inference schemes (Wilson & Nickisch, 2015; Wilson et al., 2016b). Subsequent work has explored architectural extensions, variational and amortized inference, calibration strategies, and Bayesian treatments to improve scalability or uncertainty quantification (Al-Shedivat et al., 2017; Tran et al., 2019; Dai et al., 2020; Ober et al., 2021; van Amersfoort et al., 2022; Achituve et al., 2023; Liu et al., 2023; Matias et al., 2024; Zhao et al., 2025).

Another closely related line of works place a Bayesian linear model on top of neural features, yielding a low-rank deep kernel by construction (Lazaro-Gredilla & Figueiras-Vidal, 2010; Huang et al., 2015). These include more recent Bayesian last layer (BLL) models (Snoek et al., 2015; Kristiadi et al., 2020; Watson et al., 2021; Fiedler & Lucia, 2023; Harrison et al., 2024). Compared to these works, our DBK expands the design space of deep bases and unifies sparse DKL under the same framework. This enables an in-depth analysis of training objectives and uncertainty behavior by generalizing ideas from sparse GPs.

Low-rank kernel GPs. Scalability in GP inference is commonly achieved by exploiting low-rank structure in the covariance. Sparse GP methods approximate the kernel using a small set of inducing variables, leading to Nyström-type approximations and scalable variational objectives (Williams & Seeger, 2000; Smola & Schökopf, 2000; Snelson & Ghahramani, 2005; Quiñonero-Candela & Rasmussen, 2005; Titsias, 2009; Hensman et al., 2013). These ideas have been generalized to inter-domain GPs, which define inducing variables by linear functionals (Lázaro-Gredilla & Figueiras-Vidal, 2009; Hensman et al., 2018; Dutordoir et al., 2020; Cunningham et al., 2023). Other approaches construct low-rank kernels directly using fixed or analytically derived basis functions, including relevance vector machines (Tipping, 2001), random and learned Fourier features based on Bochner's theorem (Rahimi & Recht, 2007; Lázaro-Gredilla et al., 2010; Yang et al., 2015; Dao et al., 2017; Ton et al., 2018; Li et al., 2024), and eigenfunctions (Wang et al., 2024).

D. Algorithms

We summarize the training and prediction procedures for (i) exact MML training of DBK and prediction based on the closed-form posterior, and (ii) mini-batch training of DBK via the proposed dPPGP objective and prediction under the variational formulation.

Algorithm 1 Exact DBK: Training

Input: Data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, initial deep basis $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^r$ and noise variance $\sigma_\epsilon^2 > 0$
Output: Learned $\boldsymbol{\phi}(\cdot)$ and σ_ϵ^2

1: **while** not converged **do**
2: Compute features $\boldsymbol{\phi}(\boldsymbol{x}_i)$ for $i \in [n]$.
3: Form $\boldsymbol{\Lambda}_{\boldsymbol{X}} \leftarrow \boldsymbol{\Phi}_{\boldsymbol{X}}^\top \boldsymbol{\Phi}_{\boldsymbol{X}} + \sigma_\epsilon^2 \boldsymbol{I}_r \in \mathbb{R}^{r \times r}$.
4: Evaluate log marginal likelihood (11):

$$\log p(\boldsymbol{y}) \leftarrow -\frac{n}{2}\log(2\pi) - \frac{n-r}{2}\log(\sigma_\epsilon^2) - \frac{1}{2}\log|\boldsymbol{\Lambda}_{\boldsymbol{X}}| - \frac{1}{2\sigma_\epsilon^2}\|\boldsymbol{y}\|_2^2 + \frac{1}{2\sigma_\epsilon^2}\|\boldsymbol{\Lambda}_{\boldsymbol{X}}^{-1/2}\boldsymbol{\Phi}_{\boldsymbol{X}}^\top \boldsymbol{y}\|_2^2.$$

5: Take a gradient step on $\boldsymbol{\phi}(\cdot)$ and σ_ϵ^2 to maximize $\log p(\boldsymbol{y})$.
6: **end while**

Algorithm 2 Exact DBK: Prediction

Input: Training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, deep basis $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^r$, noise variance $\sigma_\epsilon^2 > 0$, input $\boldsymbol{x}^*$
Output: Predictive mean $\hat{\mu}_f(\boldsymbol{x}^*)$ and variance $(\hat{\sigma}_f^2(\boldsymbol{x}^*) + \sigma_\epsilon^2)$.

1: Compute features $\boldsymbol{\phi}(\boldsymbol{x}_i)$ for $i \in [n]$ and $\boldsymbol{\phi}(\boldsymbol{x}^*) \in \mathbb{R}^r$.
2: Form $\boldsymbol{\Lambda}_{\boldsymbol{X}} \leftarrow \boldsymbol{\Phi}_{\boldsymbol{X}}^\top \boldsymbol{\Phi}_{\boldsymbol{X}} + \sigma_\epsilon^2 \boldsymbol{I}_r$.
3: Compute $\hat{\mu}_f(\boldsymbol{x}^*) \leftarrow \boldsymbol{\phi}(\boldsymbol{x}^*)^\top \boldsymbol{\Lambda}_{\boldsymbol{X}}^{-1} \boldsymbol{\Phi}_{\boldsymbol{X}}^\top \boldsymbol{y}$ and $\hat{\sigma}_f^2(\boldsymbol{x}^*) \leftarrow \sigma_\epsilon^2 \boldsymbol{\phi}(\boldsymbol{x}^*)^\top \boldsymbol{\Lambda}_{\boldsymbol{X}}^{-1} \boldsymbol{\phi}(\boldsymbol{x}^*)$.

Algorithm 3 DBK with dPPGP: Training

Input: Data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, initial deep basis $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^r$, noise variance $\sigma_\epsilon^2 > 0$, and variational parameters $(\boldsymbol{m}, \boldsymbol{L})$, trace regularization weight $\alpha \geq 0$, KL regularization weight $\beta \geq 0$
Output: Learned $\boldsymbol{\phi}(\cdot)$, σ_ϵ^2, and $(\boldsymbol{m}, \boldsymbol{L})$

1: **while** not converged **do**
2: Sample a mini-batch $\mathcal{B} \subset [n]$ of size b.
3: Compute features $\boldsymbol{\phi}(\boldsymbol{x}_i)$ for $i \in \mathcal{B}$.
4: Compute $\hat{\mu}_f(\boldsymbol{x}_i) \leftarrow \langle \boldsymbol{m}, \boldsymbol{\phi}(\boldsymbol{x}_i) \rangle$ and $\hat{\sigma}_f^2(\boldsymbol{x}_i) \leftarrow \|\boldsymbol{L}^\top \boldsymbol{\phi}(\boldsymbol{x}_i)\|^2$ for $i \in \mathcal{B}$.
5: Compute $\tilde{k}_b \leftarrow \max_{i \in \mathcal{B}} \|\boldsymbol{\phi}(\boldsymbol{x}_i)\|^2$ and $\mathcal{L}_{\text{trace}} \leftarrow \frac{1}{b}\sum_{i \in \mathcal{B}} \frac{\tilde{k}_b - \|\boldsymbol{\phi}(\boldsymbol{x}_i)\|^2}{2\sigma_\epsilon^2}$
6: Evaluate mini-batch objective (15):

$$\mathcal{L}_{\text{dPPGP}} \leftarrow \frac{1}{b}\sum_{i \in \mathcal{B}} -\log \mathcal{N}(y_i; \hat{\mu}_f(\boldsymbol{x}_i), \hat{\sigma}_f^2(\boldsymbol{x}_i) + \sigma_\epsilon^2) + \alpha\, \mathcal{L}_{\text{trace}} + \frac{\beta}{n}\, \mathbb{D}_{\text{KL}}(\mathcal{N}(\boldsymbol{m}, \boldsymbol{L}\boldsymbol{L}^\top) \,\|\, \mathcal{N}(0, \boldsymbol{I}_r)).$$

7: Take a gradient step on $\boldsymbol{\phi}(\cdot)$, σ_ϵ^2, and $(\boldsymbol{m}, \boldsymbol{L})$ to minimize $\mathcal{L}_{\text{dPPGP}}$.
8: **end while**

Algorithm 4 DBK with dPPGP: Prediction

Input: Deep basis $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^r$, noise variance $\sigma_\epsilon^2 > 0$, variational parameters $(\boldsymbol{m}, \boldsymbol{L})$, input $\boldsymbol{x}^*$
Output: Predictive mean $\hat{\mu}_f(\boldsymbol{x}^*)$ and variance $(\hat{\sigma}_f^2(\boldsymbol{x}^*) + \sigma_\epsilon^2)$

1: Compute feature $\boldsymbol{\phi}(\boldsymbol{x}^*)$.
2: Compute $\hat{\mu}_f(\boldsymbol{x}^*) \leftarrow \langle \boldsymbol{m}, \boldsymbol{\phi}(\boldsymbol{x}^*) \rangle$ and $\hat{\sigma}_f^2(\boldsymbol{x}^*) \leftarrow \|\boldsymbol{L}^\top \boldsymbol{\phi}(\boldsymbol{x}^*)\|_2^2$.

Table 3. **Estimated compute.** Runtime is wall-clock time on one V100 GPU.

Experiment group	Estimated GPU-hours	Typical runtime per run
1-D synthetic studies	20	2–3 min median
UCI tabular studies	150	4 min median; up to 50 min for largest datasets
Mobile internet studies	30	30 min median
Total	≈ 200	–

E. Experiment Details

We implement exact RBF and DKL GP models in GPyTorch (Gardner et al., 2018), leveraging its highly optimized BBMM algorithms. All other models, including low-rank kernel GPs and DNN, are implemented in PyTorch by ourselves. Unless otherwise stated, we use the dPPGP objective for DBK.

Compute resources. All reported experiments were run on a shared compute cluster using one NVIDIA V100 GPU per run. Each job requested 4 CPU cores and 16 GB system memory. Based on completed experiment logs, the result-generating experiments reported in this paper used approximately 200 V100 GPU-hours. This estimate includes completed runs used for reported results and may also include some repeated runs from hyperparameter sweeps; we therefore report representative per-run runtimes in addition to aggregate compute.

Network architecture. We choose $\boldsymbol{g} : \mathbb{R}^d \to \mathbb{R}^h$, the backbone of the deep basis map, to be a ResNet (He et al., 2016; Gorishniy et al., 2021). The input is first linearly projected from d to h dimensions, after which all residual blocks operate in the h-dimensional space. Each residual block applies normalization to the block input, followed by two linear layers with a point-wise nonlinearity in between, and adds the result back to the input via a residual connection. After the residual stack, a final normalization and activation function produce the h-dimensional representation. We use SiLU activations, layer normalization, and two residual blocks for all NN-based models and throughout all experiments.

Basis expansion. On top of the backbone $\boldsymbol{g}$, we consider two choices of expansion layers:

1. **DBK-SiLU**: The h-dimensional backbone output is mapped to r basis functions using a linear layer followed by a SiLU activation. The resulting basis vector is scaled element-wise by a learnable diagonal factor, initialized with random sign flips and scaled by $1/\sqrt{r}$ to stabilize the basis magnitude.
2. **DBK-RBF**: We apply an inducing-point kernel expansion to the h-dimensional backbone output. The basis is parameterized by a set of learnable inducing points $\boldsymbol{Z} \in \mathbb{R}^{r\times h}$ and kernel hyperparameters, including ARD lengthscales $\{\ell_j\}_{j=1}^h$ and a variance σ_k^2. Inducing points are initialized uniformly in $[-1, 1]^h$, and ARD lengthscales are initialized isotropically to $\ell_j = \sqrt{h}$. The RBF kernel with ARD takes the form

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_k^2 \exp\left(-\sum_{j=1}^{h} \frac{(x_j - x_j')^2}{2\ell_j^2}\right), \tag{16}$$

   which is used to construct the inducing-point basis as in (12).

Variational distribution. The variational distribution $q(\boldsymbol{w})$ is parameterized as $\mathcal{N}(\boldsymbol{m}, \boldsymbol{L}\boldsymbol{L}^\top)$, where $\boldsymbol{m} \in \mathbb{R}^r$ and $\boldsymbol{L} \in \mathbb{R}^{r\times r}$ is lower triangular with positive diagonals. We represent $\boldsymbol{m}$ directly as a learnable parameter. For the Cholesky factor, we store (i) an unconstrained vector for the log-diagonal entries and set the diagonal to $\exp(\cdot)$ to enforce positivity, and (ii) an unconstrained matrix for the off-diagonal entries, from which we keep only the strictly lower-triangular part. We then form $\boldsymbol{L}$ by adding the strictly lower-triangular part and the diagonal matrix. We initialize the log-diagonal entries with an offset of $-\frac{1}{2}\log r$ and scale the off-diagonal initialization by $1/r$ for numerical stability.

Baselines. We compare the proposed DBKs against a range of baselines:

1. **DNN**: A deterministic neural network that directly outputs a prediction using a linear layer atop the backbone $\boldsymbol{g}$, trained by minimizing the standard MSE loss.

2. **RBF** and **SV-RBF**: Exact GP regression with an RBF kernel and its scalable variant trained with the SVGP objective (5). The RBF kernel uses ARD lengthscales as in (16). For SV-RBF, inducing points are treated as learnable parameters and initialized uniformly in $[-1, 1]^d$. The ARD lengthscales are initialized isotropically to $\ell_j = \sqrt{d}$ for all dimensions, and optimized jointly with the inducing locations and kernel variance under the standard SVGP objective. We adopt the whitened formulation of SVGP (Matthews, 2017; Jankowiak et al., 2020), parameterizing the variational distribution over $\boldsymbol{w}$ same as described above rather than the inducing variables $\boldsymbol{u}$.

3. **DKL**, **SV-DKL**, and **PP-DKL**: Deep kernel learning (Wilson et al., 2016a) models that apply an RBF base kernel (16) to h-dimensional representations extracted by the backbone $\boldsymbol{g}$. DKL denotes exact GP inference in the learned latent space. Our SV-DKL baseline treats inducing points as free learnable parameters in the latent space, using the same initialization as DBK-RBF and whitened formulation. Note that this differs from the original SV-DKL paper (Wilson et al., 2016b), which places inducing points on structured grids and optimizes the GP ELBO using kernel interpolation (Wilson & Nickisch, 2015). PP-DKL further replaces the SVGP objective with the PPGP objective (6) to improve uncertainty calibration.

4. **VBLL**: Variational Bayesian last layer (Harrison et al., 2024), which places a Gaussian prior on the weights of the final linear layer of an NN and performs variational inference by maximizing the ELBO. For a controlled comparison, we construct VBLL so that the only difference from DBK-SiLU lies in the training objective. Specifically, we use the same basis map $\boldsymbol{\phi}$ as DBK-SiLU and choose a standard normal prior $\mathcal{N}(\mathbf{0}, \boldsymbol{I}_r)$ on the last-layer weights. We do not incorporate additional components from the original VBLL framework, such as structured noise covariance or hierarchical priors. The resulting VBLL objective is

$$\mathcal{L}_{\text{VBLL}} = \frac{1}{b} \sum_{(\boldsymbol{x},y)} \left\{ -\log \mathcal{N}(y; \hat{\mu}_f(\boldsymbol{x}), \sigma_\epsilon^2) + \frac{\hat{\sigma}_f^2(\boldsymbol{x})}{2\sigma_\epsilon^2} \right\} + \frac{1}{n} \mathbb{D}_{\text{KL}}(\mathcal{N}(\boldsymbol{m}, \boldsymbol{L}\boldsymbol{L}^\top) \,\|\, \mathcal{N}(\mathbf{0}, \boldsymbol{I}_r)).$$

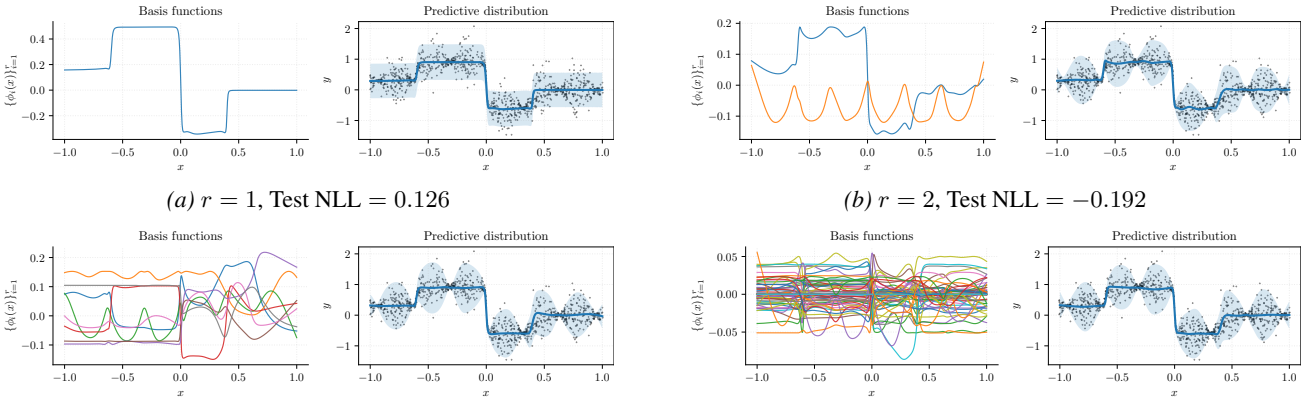
Hyperparameters. All models are trained using the AdamW optimizer with a learning rate of 10^{-3}. Weight decay with coefficient 10^{-2} is applied only to the backbone network parameters and not to kernel, variational, or noise parameters. We use a constant mean function, initialized to zero, for all GP models. The noise variance σ_ϵ^2 is initialized to 10^{-2} and constrained to be no smaller than 10^{-6} for numerical stability. Dataset-specific hyperparameters, such as the hidden size, rank and regularization strengths, are detailed in the corresponding subsections below.

Evaluation protocol. We evaluate all models using standard metrics for both predictive accuracy and uncertainty quantification. Predictive accuracy is measured by the mean absolute error (MAE), defined as the average absolute difference between predictions and ground-truth targets. Probabilistic performance is assessed using the negative log-likelihood (NLL) under the predictive Gaussian distribution and the continuous ranked probability score (CRPS), which measures the discrepancy between the predictive cumulative distribution function and the empirical distribution. For a test point with predictive density p_i and CDF F_i, including observation noise, $\text{NLL}_i = -\log p_i(y_i)$ and $\text{CRPS}_i = \int_{-\infty}^{\infty} (F_i(t) - \mathbf{1}\{t \geq y_i\})^2 \, dt$; we report both metrics averaged over test points. To further assess uncertainty calibration, we report the width of prediction intervals (PI) at a target confidence level of 95% and the empirical coverage, defined as the fraction of test targets falling within the corresponding predictive intervals. During training, we select the best model checkpoint by monitoring the validation NLL at the end of every epoch. Hyperparameters are tuned based on validation performance. All reported results correspond to the final evaluation on the held-out test set.

E.1. 1-D Synthetic Data

Data generation. We generate a 1-D regression dataset following the procedure described in Section 4.1. In total, we sample 100,000 training points, 1,000 validation points, and 1,000 test points. For the scaling experiments, we further construct smaller training sets by randomly subsampling $n \in \{500, 1{,}000, 2{,}000, 5{,}000, 10{,}000, 50{,}000\}$ points from the full training set, while keeping the validation and test sets fixed across all runs.

Hyperparameters. The hidden size is set to $h = 64$ for the backbone ResNet $\boldsymbol{g}$ in DKL and DBK. For dPPGP, we set the trace regularization and KL regularization weights to $\alpha = \beta = 0.01$. We set the rank to $r = 128$ for Figure 1 and Figures 4. To study the effect of rank, we vary the number of basis functions as $r \in \{1, 2, 4, 8, 16, 32, 64\}$, keeping all other hyperparameters fixed. For exact GP models, we perform full-batch training for 2,000 gradient steps, using the modified



(a) $r = 1$, Test NLL $= 0.126$ *(b)* $r = 2$, Test NLL $= -0.192$

(c) $r = 8$, Test NLL $= -0.215$ *(d)* $r = 64$, Test NLL $= -0.225$

Figure 9. **Additional visualizations under different ranks.** We show the basis functions and predictive distributions of DBK (dPPGP) for $r \in \{1, 2, 8, 64\}$ on the 1-D synthetic dataset. For each rank, we show the best-performing run selected by test NLL among 10 random seeds. The experiment setting is identical to that of Figure 5.

Table 4. **Additional DBK expansion layers on 1-D synthetic data.** Results are mean $\pm$ standard deviation over 10 random seeds. Lower is better.

Metric	DBK-RBF	DBK-SiLU	DBK-Fourier	DBK-Spline
MAE	.194 $\pm$ .001	.194 $\pm$ .001	.194 $\pm$ .001	.196 $\pm$ .001
NLL	-.187 $\pm$ .039	-.180 $\pm$ .033	-.183 $\pm$ .016	-.181 $\pm$ .017

conjugate gradient algorithm (Gardner et al., 2018) and the Lanczos variance estimates (LOVE) (Pleiss et al., 2018). We use up to 512 CG iterations and cap the Lanczos rank at 128.

For DBK trained with dPPGP, we use mini-batch training with batch size 200 and train for up to 2,000 epochs, employing early stopping when the validation NLL does not improve for 100 consecutive epochs.

Additional visualizations. Figure 9 provides a qualitative view of how the learned basis functions and predictive distributions evolve as the rank increases. When $r = 1$, the model collapses to a single global basis function that resembles the ground-truth mean. As the rank increases to $r = 8$, the learned bases become more diverse, enabling the model to represent both sharp transitions in the mean and the sinusoidally varying uncertainty. At $r = 64$, the basis functions are highly redundant and distributed, and the predictive distribution closely matches that of lower but sufficient ranks, indicating diminishing returns beyond moderate rank.

Additional expansion layers. The DBK construction is not restricted to the two expansion layers used in the main experiments. To provide a small check of this design space, we evaluate two additional variants on the 1-D synthetic dataset. DBK-Fourier uses

$$\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{D}[\sin(\Omega \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{b}); \cos(\Omega \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{b})],$$

where Ω is a frequency matrix, $\boldsymbol{b}$ is a phase vector, and $\boldsymbol{D}$ is a diagonal output scale. DBK-Spline uses fixed degree-p B-spline basis functions on the learned representation,

$$\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{D}[B_{1,p}(g_1(\boldsymbol{x})), \ldots, B_{q,p}(g_h(\boldsymbol{x}))]^\top.$$

We set $h = 2$, $p = 3$, and $q = r/h$, so the total rank remains r. The setting is otherwise identical to the 1-D dPPGP experiment: we use $n = 5000$ training points, batch size 200, rank 128, AdamW with learning rate 10^{-3}, and $\alpha = \beta = 0.01$. As shown in Table 4, Fourier and spline bases perform similarly to the two main DBK variants in this simple setting. This supports the claim that the DBK formulation exposes a broader basis design space, while also showing why a full architectural sweep is outside the scope of the main experiments.

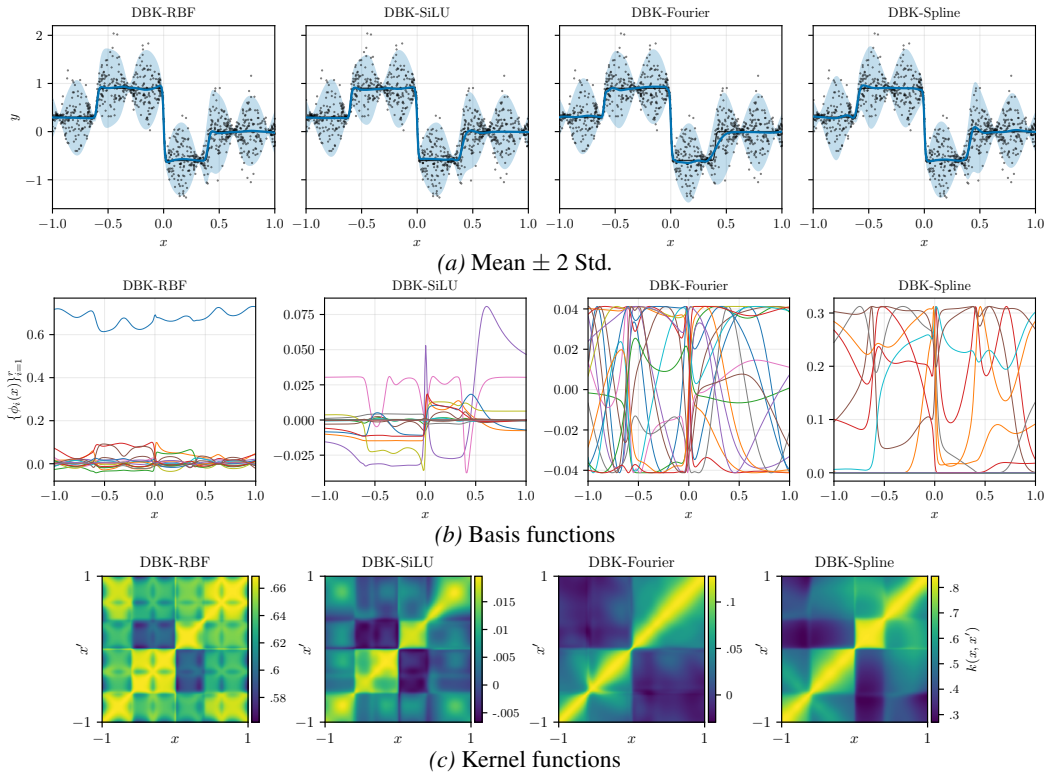


Figure 10. **Visualizations of additional DBK expansion layers.** We show predictions, learned bases, and kernel functions for DBK-RBF, DBK-SiLU, DBK-Fourier, and DBK-Spline. For each method, we show the best-performing run selected by test NLL among 10 random seeds.

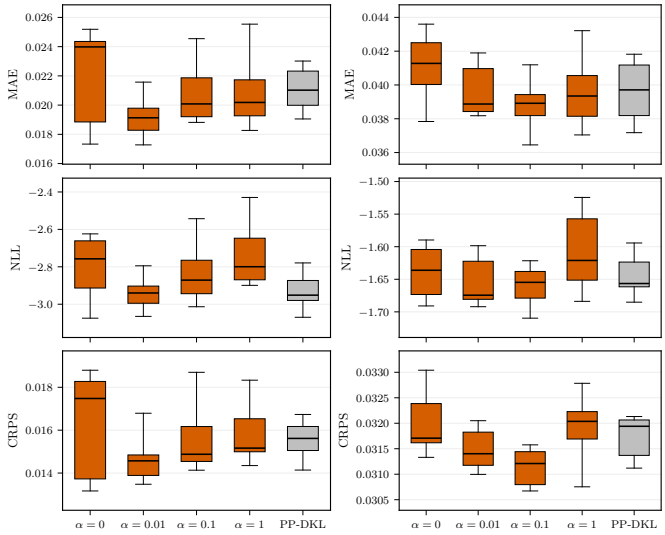
Comparison with DKL (SKI). The scaling comparison in Figure 4 labels RBF (Exact) and DKL (Exact) as exact GP baselines. To also account for the original scalable DKL mechanism, we compare against a DKL baseline using SKI/KISS-GP-style interpolation (Wilson & Nickisch, 2015). This appendix baseline uses the standard structured interpolation formulation with a learned 1-D embedding and a fixed interpolation grid of size 512, and follows the same data-generation and evaluation protocol as Figure 4. Figure 12 reports DKL (SKI) alongside the exact-GP and DBK curves for the same scaling study. DKL (SKI) extends DKL to the larger sample sizes, but its interpolation machinery introduces substantial practical overhead and its predictive performance remains worse than DBK (dPPGP).

E.2. UCI Regression

Data preprocessing. For each UCI regression dataset, we randomly split the data into training, validation, and test sets with a ratio of 8:1:1. The sample size n reported in Table 1 corresponds to the number of training samples. Input features are scaled independently to $[-1, 1]$ using min-max normalization computed from the full dataset. Targets are normalized to zero mean and unit variance. The same random split is used across all experiments.

Hyperparameters. For all UCI experiments, we set the hidden dimension of the ResNet backbone to $h = 64$ and fix the rank to $r = 128$ for all low-rank kernels. For DBK trained with dPPGP, we tune the trace and KL regularization weights α and β over the grid $\{0, 0.01, 0.1, 1\}$ based on validation NLL. Similarly, the KL regularization weight β of PP-DKL is tuned in $\{0, 0.01, 0.1, 1\}$. Models are trained for 400 epochs with mini-batch stochastic optimization using a batch size of 1024. We omit standard RBF GPs trained with SVGP or PPGP objectives for the UCI benchmarks, as they consistently underperform neural methods in our preliminary experiments.

Sensitivity of DBK-RBF to α. Because DBK-RBF uses the same inducing-point kernel construction as PP-DKL, it provides a direct way to separate the effect of dPPGP from the basis parameterization. We run this sensitivity study on Pol and Keggdir with $\alpha \in \{0, 0.01, 0.1, 1\}$. The DBK-RBF runs use the selected KL weight from the corresponding validation



(a) Pol. *(b)* Keggdir.

Figure 11. **DBK-RBF α sensitivity with PP-DKL as the baseline.** Boxplots show 10 random seeds per setting on Pol and Keggdir.

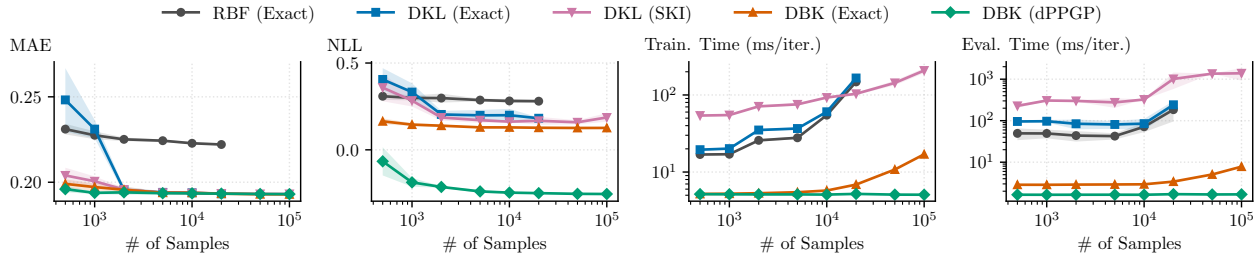


Figure 12. **Scaling behavior including DKL (SKI).** We compare DKL (SKI)/KISS-GP with RBF (Exact), DKL (Exact), DBK (Exact), and DBK (dPPGP) on the 1-D scaling study. The panels report predictive error, negative log likelihood, training time, and evaluation time under the same data-generation and evaluation protocol.

sweep ($\beta = 0$ on Pol and $\beta = 0.01$ on Keggdir), and the PP-DKL comparison uses $\beta = 0.01$. Figure 11 shows that a small positive trace weight can match or improve PP-DKL, while the gain is dataset-dependent.

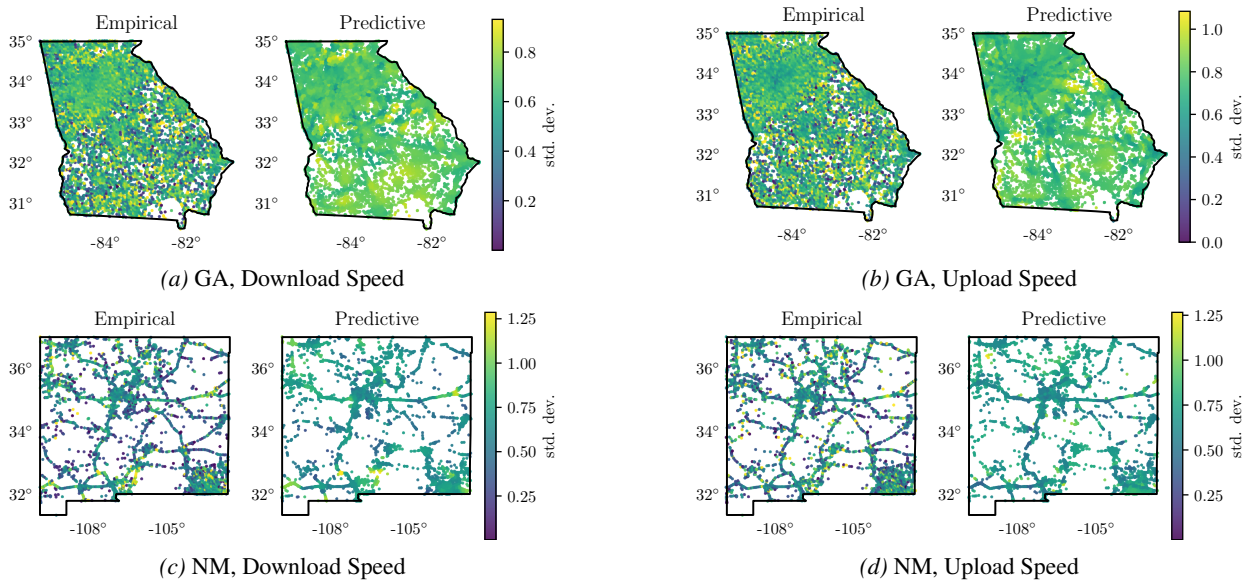
Influence of the trace term on training stability. We further check whether the in-batch trace term introduces training instability on Keggdir. We use DBK-RBF with $\alpha \in \{0, 0.01\}$, $\beta = 0.01$, and batch sizes in $\{128, 256, 512\}$. For each final checkpoint, we sample 100 mini-batches and estimate the expected trace regularizer and the gradient variance, defined as $\mathrm{tr}(\mathrm{Cov}(\boldsymbol{G}))$ where $\boldsymbol{G}$ is the full gradient vector on a random mini-batch. We also report the gradient norm averaged over the last 1,000 optimization steps. As shown in Table 5, adding the trace term can increase the final gradient variance for some batch sizes, but the gradient norms stay in a similar range and we do not observe gradient blow-up. At the same time, $\alpha = .01$ reduces the trace term by orders of magnitude and slightly improves NLL. This suggests that, in this setting, the trace term changes the variance allocation without causing unstable optimization.

E.3. Mobile Internet Quality Estimation

Data preprocessing. The dataset provides quarterly aggregated measurements of mobile internet quality on spatial tiles across the United States. Each record contains the average download or upload speed, the number of tests, and the geographic location of the tile. We focus on two states, Georgia (GA) and New Mexico (NM), and consider download and upload speeds as separate regression targets. For each state, we first filter all records whose spatial locations fall within the corresponding state boundary polygon. The spatial input $\boldsymbol{x} \in \mathbb{R}^2$ consists of longitude and latitude coordinates, which are scaled to $[-1, 1]^2$ using the bounding box of the state polygon. Targets are log-transformed (base 10) to mitigate skewness and stabilize variance. We treat each quarterly tile measurement as an independent observation, and randomly split the data into training, validation, and test sets with a ratio of 8:1:1, using a fixed random seed.

Table 5. **Gradient analysis for DBK-RBF on Keggdir.** Values are mean ± standard deviation over 10 random seeds.

Batch Size	α	Test NLL	Trace Reg.	Grad. Variance	Grad. Norm
128	0	-1.499 ± .131	4247.6 ± 3905.0	378.6 ± 183.4	39.7 ± 2.2
256	0	-1.613 ± .036	3562.0 ± 4643.5	206.5 ± 170.1	43.5 ± 3.4
512	0	-1.614 ± .051	2497.9 ± 3078.4	91.8 ± 34.1	42.7 ± 3.0
128	.01	-1.516 ± .115	1.8 ± 2.0	427.9 ± 248.7	40.2 ± 1.7
256	.01	-1.617 ± .091	1.7 ± 1.0	188.7 ± 75.5	41.3 ± 1.8
512	.01	-1.626 ± .079	1.5 ± .5	99.1 ± 25.7	43.3 ± 5.5



(a) GA, Download Speed *(b)* GA, Upload Speed

(c) NM, Download Speed *(d)* NM, Upload Speed

Figure 13. **Spatial comparison of empirical and predictive uncertainty.** For each state and target, we visualize the empirical standard deviation computed from repeated training measurements aggregated within spatial tiles of size $0.05^\circ \times 0.05^\circ$ (left) and the predictive standard deviation produced by DBK on the test set (right). Both maps share a common color scale, with the range clipped to the 99.5% percentile to reduce the influence of extreme values.

Hyperparameters. We set the hidden dimension of the ResNet backbone to $h = 256$ and fix the rank to $r = 1024$ for all methods. For DBK trained with dPPGP, we set the regularization weights $\alpha = 0.1$ and $\beta = 0.01$. Models are trained for 400 epochs with mini-batch stochastic optimization using a batch size of 1024.

Additional visualizations. Figure 13 provides a spatial comparison between empirical uncertainty estimated from repeated measurements and predictive uncertainty produced by DBK. The empirical standard deviation reflects variability in observed speeds within each spatial tile, while the predictive standard deviation captures the model's learned input-dependent uncertainty. Across both GA and NM, DBK produces smooth yet spatially varying uncertainty estimates that align well with regions exhibiting higher empirical variability, such as rural and sparsely measured areas. In contrast, more homogeneous urban regions tend to exhibit lower predictive uncertainty. These visualizations qualitatively demonstrate that DBK captures meaningful nonstationary uncertainty patterns consistent with the underlying data heterogeneity.

F. Proofs and Additional Theoretical Results

F.1. Karhunen–Loève Expansion

Theorem F.1. *Let $f \in \mathcal{X} \to \mathbb{R}$ be a zero-mean GP with kernel k satisfying the conditions of Theorem 3.1. Then, f admits the expansion*

$$f(\boldsymbol{x}) = \lim_{r\to\infty} \sum_{i=1}^{r} w_i \sqrt{\lambda_i}, \psi_i(\boldsymbol{x}), \quad w_i \overset{\text{iid}}{\sim} \mathcal{N}(0,1). \tag{17}$$

The series converges almost surely.

Proof. Let ν be a Borel probability measure on compact $\mathcal{X}$ and define the covariance operator

$$(\boldsymbol{C}g)(\boldsymbol{x}) = \int_{\mathcal{X}} k(\boldsymbol{x}, \boldsymbol{x}')\, g(\boldsymbol{x}')\, d\nu(\boldsymbol{x}').$$

By the assumptions of Theorem 3.1, $\boldsymbol{C}$ is self-adjoint, positive, and trace-class on $L^2(\nu)$ with eigenpairs $\{(\lambda_i, \psi_i)\}_{i\geq 1}$ forming an orthonormal basis, and Mercer gives

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{i=1}^{\infty} \lambda_i \psi_i(\boldsymbol{x}) \psi_i(\boldsymbol{x}') \quad \text{uniformly on } \mathcal{X} \times \mathcal{X}$$

Let $w_i \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$ and set $f_r(\boldsymbol{x}) = \sum_{i=1}^{r} \sqrt{\lambda_i}\, w_i\, \psi_i(\boldsymbol{x})$. Then $\mathbb{E}[f_r(\boldsymbol{x})] = 0$ and

$$\mathrm{Cov}(f_r(\boldsymbol{x}), f_r(\boldsymbol{x}')) = \sum_{i=1}^{r} \lambda_i \psi_i(\boldsymbol{x}) \psi_i(\boldsymbol{x}') \to k(\boldsymbol{x}, \boldsymbol{x}').$$

Thus for any $\boldsymbol{X} = \{\boldsymbol{x}\}_{i=1}^{n} \subset \mathcal{X}$, $[f_r(\boldsymbol{x}_1), \ldots, f_r(\boldsymbol{x}_n)]^\top \xrightarrow{d} \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}})$, so the limit has the same finite-dimensional distributions as f. Moreover $\sum_i \lambda_i = \mathrm{tr}(\boldsymbol{C}) = \int_{\mathcal{X}} k(\boldsymbol{x}, \boldsymbol{x})\, d\nu(\boldsymbol{x}) < \infty$, hence $\sum_i \lambda_i w_i^2 < \infty$ a.s., implying $f_r \to f$ in $L^2(\nu)$ a.s., i.e.,

$$f(\boldsymbol{x}) = \lim_{r\to\infty} \sum_{i=1}^{r} w_i \sqrt{\lambda_i}\, \psi_i(\boldsymbol{x}) \quad \text{a.s.}$$

□

F.2. Exact DBK via MML

Log marginal likelihood. Denote $\boldsymbol{\Sigma}_{\boldsymbol{X}} = \boldsymbol{\Phi}_{\boldsymbol{X}} \boldsymbol{\Phi}_{\boldsymbol{X}}^\top + \sigma_\epsilon^2 \boldsymbol{I}_n$ and $\boldsymbol{\Lambda}_{\boldsymbol{X}} = \boldsymbol{\Phi}_{\boldsymbol{X}}^\top \boldsymbol{\Phi}_{\boldsymbol{X}} + \sigma_\epsilon^2 \boldsymbol{I}_r$. The log marginal likelihood

$$\log p(\boldsymbol{y}) = \log \mathcal{N}(\boldsymbol{y}; \boldsymbol{0}, \boldsymbol{\Sigma}_{\boldsymbol{X}}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{\Sigma}_{\boldsymbol{X}}| - \frac{1}{2}\boldsymbol{y}^\top \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1} \boldsymbol{y}$$

decomposes into a complexity penalty and a data fit term. By the Sherman–Morrison–Woodbury formula,

$$\boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1} = \frac{1}{\sigma_\epsilon^2}\big(\boldsymbol{I}_n - \boldsymbol{\Phi}_{\boldsymbol{X}} \boldsymbol{\Lambda}_{\boldsymbol{X}}^{-1} \boldsymbol{\Phi}_{\boldsymbol{X}}^\top\big), \tag{18}$$

which yields

$$-\frac{1}{2}\boldsymbol{y}^\top \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1} \boldsymbol{y} = -\frac{1}{2\sigma_\epsilon^2}\|\boldsymbol{y}\|_2^2 + \frac{1}{2\sigma_\epsilon^2}\|\boldsymbol{\Lambda}_{\boldsymbol{X}}^{-1/2} \boldsymbol{\Phi}_{\boldsymbol{X}}^\top \boldsymbol{y}\|_2^2.$$

By the matrix determinant lemma,

$$|\boldsymbol{\Sigma}_{\boldsymbol{X}}| = \sigma_\epsilon^{2(n-r)} |\boldsymbol{\Lambda}_{\boldsymbol{X}}|,$$

and thus

$$-\frac{1}{2}\log|\boldsymbol{\Sigma}_{\boldsymbol{X}}| = -\frac{n-r}{2}\log(\sigma_\epsilon^2) - \frac{1}{2}\log|\boldsymbol{\Lambda}_{\boldsymbol{X}}|.$$

Combining terms gives (11).

Gradient of log marginal likelihood. By the chain rule,

$$\nabla_{\boldsymbol{\Sigma}_{\boldsymbol{X}}} \log p(\boldsymbol{y}) = -\frac{1}{2}\boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1} + \frac{1}{2}\boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1} \boldsymbol{y}\boldsymbol{y}^\top \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1},$$

and

$$\nabla_{\boldsymbol{\Phi}_{\boldsymbol{X}}} \log p(\boldsymbol{y}) = 2(\nabla_{\boldsymbol{\Sigma}_{\boldsymbol{X}}} \log p(\boldsymbol{y}))\boldsymbol{\Phi}_{\boldsymbol{X}} = -\boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}\boldsymbol{\Phi}_{\boldsymbol{X}} + \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}\boldsymbol{y}\boldsymbol{y}^\top \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}\boldsymbol{\Phi}_{\boldsymbol{X}}.$$

The required solves are computed via (18):

$$[\boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}\boldsymbol{\Phi}_{\boldsymbol{X}}, \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}\boldsymbol{y}] = \frac{1}{\sigma_\epsilon^2}\Big([\boldsymbol{\Phi}_{\boldsymbol{X}}, \boldsymbol{y}] - \boldsymbol{\Phi}_{\boldsymbol{X}} \boldsymbol{\Lambda}_{\boldsymbol{X}}^{-1} \boldsymbol{\Phi}_{\boldsymbol{X}}^\top [\boldsymbol{\Phi}_{\boldsymbol{X}}, \boldsymbol{y}]\Big).$$

F.3. Proof of Theorem 3.2

Under the model $p(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{0}, \boldsymbol{\Sigma}_{\boldsymbol{X}})$ with $\boldsymbol{\Sigma}_{\boldsymbol{X}} \succ 0$,

$$\log p(\boldsymbol{y}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{\Sigma}_{\boldsymbol{X}}| - \frac{1}{2}\,\boldsymbol{y}^\top \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}\boldsymbol{y}.$$

Taking expectation under $p_{\mathrm{gt}}(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{\mu}_{\mathrm{gt}}, \boldsymbol{D}_{\mathrm{gt}})$ gives

$$\mathbb{E}_{p_{\mathrm{gt}}(\boldsymbol{y})}[\log p(\boldsymbol{y})] = \text{const} - \frac{1}{2}\log|\boldsymbol{\Sigma}_{\boldsymbol{X}}| - \frac{1}{2}\,\mathbb{E}_{p_{\mathrm{gt}}(\boldsymbol{y})}\left[\boldsymbol{y}^\top \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}\boldsymbol{y}\right].$$

Using $\mathbb{E}[\boldsymbol{y}^\top \boldsymbol{A}\boldsymbol{y}] = \mathrm{tr}(\boldsymbol{A}\,\mathbb{E}[\boldsymbol{y}\boldsymbol{y}^\top])$ (for any symmetric $\boldsymbol{A}$),

$$\mathbb{E}_{p_{\mathrm{gt}}(\boldsymbol{y})}\left[\boldsymbol{y}^\top \boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}\boldsymbol{y}\right] = \mathrm{tr}\Big(\boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}\,\mathbb{E}_{p_{\mathrm{gt}}(\boldsymbol{y})}[\boldsymbol{y}\boldsymbol{y}^\top]\Big).$$

Moreover, since p_{gt} is Gaussian,

$$\mathbb{E}_{p_{\mathrm{gt}}(\boldsymbol{y})}[\boldsymbol{y}\boldsymbol{y}^\top] \;=\; \boldsymbol{\mu}_{\mathrm{gt}}\boldsymbol{\mu}_{\mathrm{gt}}^\top + \boldsymbol{D}_{\mathrm{gt}}.$$

Therefore,

$$\mathbb{E}_{p_{\mathrm{gt}}(\boldsymbol{y})}[\log p(\boldsymbol{y})] = \text{const} - \frac{1}{2}\log|\boldsymbol{\Sigma}_{\boldsymbol{X}}| - \frac{1}{2}\;\mathrm{tr}\Big(\boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}(\boldsymbol{\mu}_{\mathrm{gt}}\boldsymbol{\mu}_{\mathrm{gt}}^\top + \boldsymbol{D}_{\mathrm{gt}})\Big). \tag{19}$$

Reparameterize by $\boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1} \succ 0$. Since $-\log|\boldsymbol{\Sigma}_{\boldsymbol{X}}| = \log|\boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}|$, the objective is a concave function of $\boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}$ (log-det is strictly concave; trace is linear), so this is a convex optimization problem. The first-order condition is sufficient and necessary. Differentiating w.r.t. $\boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}$ yields

$$\nabla_{\boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}}\,\mathbb{E}_{p_{\mathrm{gt}}(\boldsymbol{y})}[\log p(\boldsymbol{y})] = \frac{1}{2}\,\boldsymbol{\Sigma}_{\boldsymbol{X}} - \frac{1}{2}\big(\boldsymbol{\mu}_{\mathrm{gt}}\boldsymbol{\mu}_{\mathrm{gt}}^\top + \boldsymbol{D}_{\mathrm{gt}}\big).$$

Setting the gradient to zero gives the unique maximizer

$$\boldsymbol{\Sigma}_{\boldsymbol{X}}^{*} = \boldsymbol{\mu}_{\mathrm{gt}}\boldsymbol{\mu}_{\mathrm{gt}}^\top + \boldsymbol{D}_{\mathrm{gt}} = \mathbb{E}_{p_{\mathrm{gt}}(\boldsymbol{y})}[\boldsymbol{y}\boldsymbol{y}^\top].$$

F.4. Proof of Theorem 3.4

By (19), maximizing $\mathbb{E}_{p_{\mathrm{gt}}(\boldsymbol{y})}[\log p(\boldsymbol{y})]$ is equivalent to minimize

$$\mathcal{J}(\boldsymbol{\Sigma}_{\boldsymbol{X}}) = \log|\boldsymbol{\Sigma}_{\boldsymbol{X}}| + \mathrm{tr}(\boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}\boldsymbol{S}_{\boldsymbol{X}})$$

over the feasible set

$$\boldsymbol{\Sigma}_{\boldsymbol{X}} = \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma_\epsilon^2\boldsymbol{I}_n, \qquad \boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} \succeq 0,\ \mathrm{rank}(\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}) \le r,\ \sigma_\epsilon^2 > 0,$$

Fix any feasible $\boldsymbol{\Sigma}_{\boldsymbol{X}} \succ 0$ and write its eigendecomposition $\boldsymbol{\Sigma}_{\boldsymbol{X}} = \boldsymbol{V}\,\mathrm{diag}(s_1, \ldots, s_n)\boldsymbol{V}^\top$ with $s_1 \ge \cdots \ge s_n > 0$. Then $\boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1} = \boldsymbol{V}\,\mathrm{diag}(s_1^{-1}, \ldots, s_n^{-1})\boldsymbol{V}^\top$. For fixed eigenvalues (s_i), the term $\log|\boldsymbol{\Sigma}_{\boldsymbol{X}}| = \sum_i \log s_i$ is fixed, and we only need to minimize $\mathrm{tr}(\boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}\boldsymbol{S}_{\boldsymbol{X}})$ over $\boldsymbol{V}$. By von Neumann's trace inequality, the minimum of $\mathrm{tr}(\boldsymbol{\Sigma}_{\boldsymbol{X}}^{-1}\boldsymbol{S}_{\boldsymbol{X}})$ is achieved when $\boldsymbol{\Sigma}_{\boldsymbol{X}}$ and $\boldsymbol{S}_{\boldsymbol{X}}$ share eigenvectors $\boldsymbol{V} = \boldsymbol{U}$. Hence we may restrict w.l.o.g. to $\boldsymbol{\Sigma}_{\boldsymbol{X}} = \boldsymbol{U}\,\mathrm{diag}(s_1, \ldots, s_n)\boldsymbol{U}^\top$.

Since $\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} = \boldsymbol{\Sigma}_{\boldsymbol{X}} - \sigma_\epsilon^2\boldsymbol{I}_n$ has rank at most r and is PSD, $\boldsymbol{\Sigma}_{\boldsymbol{X}}$ must have the form $s_1 \ge \cdots \ge s_r \ge \sigma_\epsilon^2$ and $s_{r+1} = \cdots = s_n = \sigma_\epsilon^2$. The objective becomes

$$\mathcal{J}(\boldsymbol{\Sigma}_{\boldsymbol{X}}) = \sum_{i=1}^{r}\left(\log s_i + \frac{\lambda_i}{s_i}\right) + \sum_{i=r+1}^{n}\left(\log\sigma_\epsilon^2 + \frac{\lambda_i}{\sigma_\epsilon^2}\right).$$

For each $i \le r$, the term $\log s_i + \lambda_i/s_i$ as a function of s_i is minimized at $s_i^* = \lambda_i$ on $(0, \infty)$. Plugging $s_i = \lambda_i$, $i = 1, \ldots, r$ into $\mathcal{J}$, the only σ_ϵ^2-dependent part is

$$(n-r)\log\sigma_\epsilon^2 + \frac{1}{\sigma_\epsilon^2}\sum_{i=r+1}^{n}\lambda_i,$$

which has a unique minimizer

$$\sigma_\epsilon^{*2} = \frac{1}{n-r} \sum_{i=r+1}^{n} \lambda_i.$$

Since $s_i^* \geq \sigma_\epsilon^{*2}$, $i = 1, \ldots, r$, the solution is feasible. Hence, the optimal kernel matrix is

$$\begin{aligned} \boldsymbol{K}_{\boldsymbol{XX}}^* &= \boldsymbol{U} \operatorname{diag}(\lambda_1, \ldots, \lambda_r, \sigma_\epsilon^{*2}, \ldots, \sigma_\epsilon^{*2}) \boldsymbol{U}^\top - \sigma_\epsilon^{*2} \boldsymbol{I}_n \\ &= \sum_{i=1}^{r} (\lambda_i - \sigma_\epsilon^{*2}) \boldsymbol{u}_i \boldsymbol{u}_i^\top, \end{aligned}$$

Note that $\boldsymbol{S}_{\boldsymbol{X}} = \boldsymbol{\mu}_{\mathrm{gt}} \boldsymbol{\mu}_{\mathrm{gt}}^\top + \boldsymbol{D}_{\mathrm{gt}}$ is a rank-1 PSD update of the diagonal matrix $\boldsymbol{D}_{\mathrm{gt}}$. Let $d_1 \geq \cdots \geq d_n$ be the diagonal entries of $\boldsymbol{D}_{\mathrm{gt}}$ sorted in decreasing order. For a rank-1 Hermitian update, eigenvalues interlace (Horn & Johnson, 2012):

$$d_i \geq \lambda_{i+1} \geq d_{i+1}, \quad i = 1, \ldots, n-1.$$

F.5. Derivation of Trace Regularization

The trace regularization originally comes from the variational treatment to sparse GPs (Titsias, 2009). Here, we provide a weight-space derivation with a general basis map $\boldsymbol{\phi}$.

Assume that the low-rank kernel matrix $\boldsymbol{K}_{\boldsymbol{XX}} = \boldsymbol{\Phi}_{\boldsymbol{X}} \boldsymbol{\Phi}_{\boldsymbol{X}}^\top$ approximates a full kernel matrix $\tilde{\boldsymbol{K}}_{\boldsymbol{XX}}$ with PSD error $\tilde{\boldsymbol{K}}_{\boldsymbol{XX}} - \boldsymbol{\Phi}_{\boldsymbol{X}} \boldsymbol{\Phi}_{\boldsymbol{X}}^\top \succ 0$. Given $\boldsymbol{w}$, the conditional likelihood under $\boldsymbol{K}_{\boldsymbol{XX}}$ and $\tilde{\boldsymbol{K}}_{\boldsymbol{XX}}$ can be respectively written as

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{w}) &= \mathcal{N}(\boldsymbol{y}; \boldsymbol{\Phi}_{\boldsymbol{X}} \boldsymbol{w}, \sigma_\epsilon^2 \boldsymbol{I}_n), \\ \tilde{p}(\boldsymbol{y}|\boldsymbol{w}) &= \mathcal{N}(\boldsymbol{y}; \boldsymbol{\Phi}_{\boldsymbol{X}} \boldsymbol{w}, \tilde{\boldsymbol{K}}_{\boldsymbol{XX}} - \boldsymbol{\Phi}_{\boldsymbol{X}} \boldsymbol{\Phi}_{\boldsymbol{X}}^\top + \sigma_\epsilon^2 \boldsymbol{I}_n). \end{aligned}$$

We have

$$\begin{aligned} \log \tilde{p}(\boldsymbol{y}|\boldsymbol{w}) &\geq \mathbb{E}_{\mathcal{N}(\boldsymbol{f}; \boldsymbol{\Phi}_{\boldsymbol{X}} \boldsymbol{w}, \tilde{\boldsymbol{K}}_{\boldsymbol{XX}} - \boldsymbol{\Phi}_{\boldsymbol{X}} \boldsymbol{\Phi}_{\boldsymbol{X}}^\top)} [\log \mathcal{N}(\boldsymbol{y}; \boldsymbol{f}, \sigma_\epsilon^2 \boldsymbol{I}_n)] \\ &= \log \mathcal{N}(\boldsymbol{y}; \boldsymbol{\Phi}_{\boldsymbol{X}} \boldsymbol{w}, \sigma_\epsilon^2 \boldsymbol{I}_n) - \frac{1}{2\sigma_\epsilon^2} \operatorname{tr}(\tilde{\boldsymbol{K}}_{\boldsymbol{XX}} - \boldsymbol{\Phi}_{\boldsymbol{X}} \boldsymbol{\Phi}_{\boldsymbol{X}}^\top) \\ &= \log p(\boldsymbol{y}|\boldsymbol{w}) - \frac{1}{2\sigma_\epsilon^2} \operatorname{tr}(\tilde{\boldsymbol{K}}_{\boldsymbol{XX}} - \boldsymbol{\Phi}_{\boldsymbol{X}} \boldsymbol{\Phi}_{\boldsymbol{X}}^\top) \\ &= \log p(\boldsymbol{y}|\boldsymbol{w}) - \sum_{\boldsymbol{x}} \frac{\tilde{k}(\boldsymbol{x}, \boldsymbol{x}) - \|\boldsymbol{\phi}(\boldsymbol{x})\|^2}{2\sigma_\epsilon^2}. \end{aligned}$$

The first inequality is Jensen's inequality applied to $\log \mathbb{E}[\mathcal{N}(\boldsymbol{y}; \boldsymbol{f}, \sigma_\epsilon^2 \boldsymbol{I}_n)]$, using the concavity of the logarithm under $\boldsymbol{f} \sim \mathcal{N}(\boldsymbol{\Phi}_{\boldsymbol{X}} \boldsymbol{w}, \tilde{\boldsymbol{K}}_{\boldsymbol{XX}} - \boldsymbol{\Phi}_{\boldsymbol{X}} \boldsymbol{\Phi}_{\boldsymbol{X}}^\top)$. The bound is tight when $\tilde{p}(\boldsymbol{f}|\boldsymbol{w}) = \tilde{p}(\boldsymbol{f}|\boldsymbol{w}, \boldsymbol{y})$. Marginalizing over $\boldsymbol{w}$, we obtain the bound in (3):

$$\begin{aligned} \log \tilde{p}(\boldsymbol{y}) &= \log \mathbb{E}_{p(\boldsymbol{w})} [\tilde{p}(\boldsymbol{y}|\boldsymbol{w})] \\ &\geq \log \mathbb{E}_{p(\boldsymbol{w})} [p(\boldsymbol{y}|\boldsymbol{w})] - \sum_{\boldsymbol{x}} \frac{\tilde{k}(\boldsymbol{x}, \boldsymbol{x}) - \|\boldsymbol{\phi}(\boldsymbol{x})\|^2}{2\sigma_\epsilon^2} \\ &= \log p(\boldsymbol{y}) - \sum_{\boldsymbol{x}} \frac{\tilde{k}(\boldsymbol{x}, \boldsymbol{x}) - \|\boldsymbol{\phi}(\boldsymbol{x})\|^2}{2\sigma_\epsilon^2}, \end{aligned}$$

When the sum is substituted with an in-batch sample average, this gives the trace regularizer:

$$\mathcal{L}_{\text{trace}} = \frac{1}{b} \sum_{\boldsymbol{x}} \frac{\tilde{k}(\boldsymbol{x}, \boldsymbol{x}) - \|\boldsymbol{\phi}(\boldsymbol{x})\|^2}{2\sigma_\epsilon^2}.$$

Note that this term is included in $\mathcal{L}_{\text{SVGP}}$ (5) via $\frac{\tilde{\sigma}_f^2(\boldsymbol{x})}{2\sigma_\epsilon^2}$. For a non-inducing-point basis map $\boldsymbol{\phi}$, the full kernel $\tilde{k}$ is typically not defined. To this end, we further substitute $\tilde{k}(\boldsymbol{x}, \boldsymbol{x})$ with an in-batch maximum $\tilde{k}_b = \max \|\boldsymbol{\phi}(\boldsymbol{x})\|^2$, resulting in the decoupled trace regularizer in (15):

$$\mathcal{L}_{\text{trace}} = \frac{1}{b} \sum_{\boldsymbol{x}} \frac{\tilde{k}_b - \|\boldsymbol{\phi}(\boldsymbol{x})\|^2}{2\sigma_\epsilon^2}.$$