FedBNR: a Fully Global Federated Gaussian Process

Haolin Yu[1,2], Kaiyang Guo[3], Mahdi Karami[1†], Guojun Zhang[4†], Hongliang Li[3], Xi Chen[3], Pascal Poupart[1,2*]

[1*]University of Waterloo, Waterloo, ON, Canada.
[2]Vector Institute for AI, Toronto, ON, Canada.
[3]Noah's Ark Lab, Huawei Technologies.
[4]MiniMax, China.

*Corresponding author(s). E-mail(s): ppoupart@uwaterloo.ca;
Contributing authors: h89yu@uwaterloo.ca; guokaiyang@huawei.com;
mahdikaramia@gmail.com; guojun.zhang@uwaterloo.ca;
hongliang.li2@huawei.com; xi.chen11@mcgill.ca;
†The work was done at their previous affiliation, Huawei Noah's Ark Lab.

Abstract

Uncertainty estimation plays a key role in many practical areas such as simulation and parameter optimization. Gaussian process (GP) is one popular model that provides naturally well-calibrated uncertainty estimates. However, it is challenging to learn a global GP posterior under the federated learning (FL) framework. In FL, clients' private data should not be shared, but merging local kernels directly leads to privacy leakage. Previous works that consider federated GPs avoid it and focus on the personalized setting. This sacrifices information from other clients that can be exploited to benefit generalization. We present Federated Bayesian Neural Regression (FedBNR) that learns a global federated GP while respecting clients' privacy. We incorporate deep kernel learning and random features by defining a unifying random kernel (URK). URK enables a principled approach of learning a global posterior as if all client data is centralized. Experiments conducted on real world regression datasets show statistically significant improvements.

Keywords: federated learning, Gaussian process, random feature, Bayesian method

1 Introduction

Uncertainty estimation has been an indispensable part of many practical areas. For example, in simulations widely used for the development of aircraft, trains, and vehicles (Danquah et al., 2020) and the testing of safeguarding autonomous vehicles (Riedmaier et al., 2020), uncertainty estimation is needed to assess the uncertainty inherent in any simulation model (Riedmaier et al., 2021). Researchers have also been using Bayesian optimization to speed up material design with nanoparticles (Xia et al., 2023; Iyer et al., 2020, 2019), where uncertainty estimation is used to balance exploitation and exploration. In machine-assisted decision making (Begoli et al., 2019), it provides the trust needed to inform decisions.

Gaussian process (GP) has been an especially popular model that provides uncertain estimates (Verrelst et al., 2013; Li et al., 2021). GPs with deep kernel learning (Wilson et al., 2016b,a) provide a good balance between expressiveness and simplicity to represent model uncertainty. At one end of the spectrum, most traditional models such as traditional neural networks do not capture any uncertainty, but are simple and scalable. At the other end of the spectrum, Bayesian neural networks express a full distribution over all weights of neural networks, but exact inference tends to be infeasible. In between, a GP with a deep kernel consists of a deep neural feature extractor with a distribution over the last layer that facilitates exact inference. Since the last layer is the most important for prediction, representing its uncertainty is often sufficient to capture most of the uncertainty of a model (Gawlikowski et al., 2022; Kristiadi et al., 2020).

However, extending GPs to the federated learning (FL) setting itself is challenging. In FL, we seek to train a model in a distributed way across several clients without any data leaving the clients to preserve privacy. Several works have explored distributed GPs (Deisenroth and Ng, 2015; Yin et al., 2020) and federated GPs (Achituve et al., 2021; Kontoudis and Stilwell, 2023, 2024). While distributed GPs are designed to improve scalability, they pose an important privacy risk since sharing kernels either implies sharing data or sharing pairwise data similarity. In contrast, federated GPs including pFedGP (Achituve et al., 2021),DEC-apx-GP (Kontoudis and Stilwell, 2023), and DEC-gapx-GP (Kontoudis and Stilwell, 2024) share only the hyperparameters of kernels while learning local GP posteriors that are never shared. This personalized approach reduces privacy risks, but the local GPs do not benefit from other client information beyond the shared kernel hyperparameters. Thus, its accuracy suffers when generalization is desired or new clients come without training data. For example, a hospital that only collected chest X-rays of lung emphysema patients may also want their model to help diagnose rib fractures with the same type of input. However, personalization would prevent the model from generalizing to other ranges. Moreover, new clients with few training data cannot effectively train predictors with local GPs, even if they do not need generalization.

To address such challenges, we propose a new federated GP technique called Federated Bayesian Neural Regression (FedBNR) that learns a global federated GP. We first define a unifying random kernel (URK) which allows random feature approximation of both stationary and non-stationary kernels. FedBNR with URK avoids kernel sharing by working directly in the feature space and sharing scatter matrices, reducing

privacy risks and decreasing the computational complexity to linear in the amount of data. Since FedBNR maintains a global posterior, it makes full use of different clients' information, and performs better for generalization and new clients. The approach is demonstrated on real world regression datasets where we achieve statistically significant improvements over prior techniques both in terms of point and uncertainty estimation. We further extend our algorithm with differential privacy to provide formal privacy guarantees. The main contributions are:

- New random feature approximations of non-stationary kernels enabled by URK, and their concentration guarantees.
- A new federated GP technique FedBNR, which achieves better generalization performance in both accuracy and uncertainty calibration due to its global posterior.

The paper is structured as follows. In Sec. 2, we review essential background knowledge of GP, FL, and related works. In Sec. 3, we introduce URK and its theoretical guarantees, propose FedBNR based on URK, and extend FedBNR with differential privacy (DP). Sec. 4 details synthetic and real-world experiments. Finally, Sec. 5 concludes the work.

2 Background

Notation. We use the following notation throughout the paper. $\mathbf{X}, \mathbf{X}' \in \mathbb{R}^{p\times n}$, $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^p$ are input matrices or vectors. n is the number of data points and p is the number of features. $\mathbf{y} \in \mathbb{R}^n$ is the target vector. $\mathbf{x}_*, y_*$ indicate vectors that have not been seen by the models. $\sigma \in \mathbb{R}^+$ is the standard deviation of the additive Gaussian noise in the models we mention later. $\mathbf{I}$ is the identity matrix and $\mathbf{0}$ is the zero vector. Their dimensions can be inferred from the context. $\phi : \mathbb{R}^p \to \mathbb{R}^d$ denotes a d dimensional basis function, and $\mathbf{\Phi} = \phi(\mathbf{X})$ denotes the corresponding features of $\mathbf{X}$. $k : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ denotes a kernel function, and $\mathbf{K} = k(\mathbf{X}, \mathbf{X})$ is the corresponding kernel Gram matrix. $\mathbf{w} \in \mathbb{R}^d$ are the weights of the linear layer in a Bayesian linear regression model. $\lambda \in \mathbb{R}^+$ is the prior standard deviation on $\mathbf{w}$. $\mathbb{E}[\cdot], \mathrm{Cov}(\cdot,\cdot)$, and $\mathrm{Tr}(\cdot)$ are the expectation, covariance, and trace function. Any symbol with a c subscript is a local version of the original symbol, held by client c.

2.1 Gaussian process and random Fourier feature

A GP can be viewed as an infinite dimensional Gaussian distribution over function objects $f(\cdot)$. If restricted to a finite set of points of interest $\mathbf{X}$ on the support, a GP becomes a multi-dimensional Gaussian distribution $f(\mathbf{X})$, providing distributional information at these places. Formally, GP regression with additive Gaussian noise is established as $y = f(\mathbf{x}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. After a prior over $f(\cdot)$ is specified, the likelihood, posterior, and prediction can be computed as follows:

$$\text{Prior:} \qquad f(\cdot) \sim \mathcal{GP}(\mathbf{0}, k(\cdot,\cdot)) \tag{1}$$

$$\text{Likelihood:} \qquad (\mathbf{y}|\mathbf{X}, f(\cdot)) \sim \mathcal{N}(f(\mathbf{X}), \sigma^2\mathbf{I}) \tag{2}$$

$$\text{Posterior:} \qquad (f(\cdot)|\mathbf{X}, \mathbf{y}) \sim \mathcal{GP}(\bar{m}(\cdot), k'(\cdot,\cdot)), \text{ where} \tag{3}$$

$$\bar{m}(\cdot) = k(\cdot, \mathbf{X})(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}, \tag{4}$$

$$k'(\cdot, \cdot) = k(\cdot, \cdot) - k(\cdot, \mathbf{X})(\mathbf{K} + \sigma^2\mathbf{I})^{-1}k(\mathbf{X}, \cdot) \tag{5}$$

Prediction: $$(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) \sim \mathcal{N}(\bar{m}(\mathbf{x}_*), \sigma^2 + k'(\mathbf{x}_*, \mathbf{x}_*)) \tag{6}$$

The runtime complexity of GP is $O(n^3)$ due to $\mathbf{K}^{-1}$. Thus in practice, full GPs are often infeasible, and approximations are needed for scalability. The performance of GP models highly depends on the kernel function $k(\cdot, \cdot)$, and hyperparameters of this function can be learned by maximizing $\log \Pr_{\text{GP}}(\mathbf{y}|\mathbf{X}) = -\mathbf{y}^\top(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{y} - \log|\mathbf{K}+\sigma^2\mathbf{I}|$.

A valid kernel function is any positive definite function and can always be decomposed into the inner product of some basis functions $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top\phi(\mathbf{x}')$ (Cuturi, 2009). An important advantage of working in the function space (i.e., with kernels) is that kernel functions can correspond to infinite features (i.e., the corresponding $\phi(\mathbf{x})$ must map $\mathbf{x}$ to infinitely many linearly independent features) without paying a price in complexity. Popular kernels, such as the Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-||\mathbf{x} - \mathbf{x}'||^2/2\gamma^2)$ ($\gamma \in \mathbb{R}^+$ is a kernel hyperparameter), tend to induce infinite dimensional features, providing significant model capacity.

If the kernel has finite features (i.e. $\exists\phi(\mathbf{x})$ that maps $\mathbf{x}$ into finitely many features), a GP degenerates into Bayesian linear regression (BLR) in the feature space with parametrized model $f(\cdot) = \mathbf{w}^\top\phi(\cdot)$. Given a prior over $\mathbf{w}$, the likelihood, posterior, and prediction are:

Prior: $$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \lambda^2\mathbf{I}) \tag{7}$$

Likelihood: $$(\mathbf{y}|\mathbf{X}, \mathbf{w}) \sim \mathcal{N}(\mathbf{w}^\top\mathbf{\Phi}, \sigma^2\mathbf{I}) \tag{8}$$

Posterior: $$(\mathbf{w}|\mathbf{X}, \mathbf{y}) \sim \mathcal{N}(\bar{\mathbf{w}}, \mathbf{A}^{-1}), \text{ where} \tag{9}$$

$$\bar{\mathbf{w}} = \sigma^{-2}\mathbf{A}^{-1}\mathbf{\Phi}\mathbf{y} \text{ and } \mathbf{A} = \sigma^{-2}\mathbf{\Phi}\mathbf{\Phi}^\top + \lambda^{-2}\mathbf{I} \tag{10}$$

Prediction: $$(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) \sim \mathcal{N}(\phi(\mathbf{x}_*)\bar{\mathbf{w}}, \sigma^2 + \phi(\mathbf{x}_*)^\top\mathbf{A}^{-1}\phi(\mathbf{x}_*)) \tag{11}$$

Recall $\phi(\mathbf{X}) \in \mathbb{R}^d$. Then the complexity of BLR is $O(nd^2 + d^3)$ due to $\mathbf{A}^{-1}$. Hyperparameters of ϕ can be learned similarly by maximizing $\log \Pr_{\text{BLR}}(\mathbf{y}|\mathbf{X}) = -n\log\sigma^2 - \log|\lambda^2\mathbf{I}| - \log|\mathbf{A}| - ||\mathbf{y} - \mathbf{\Phi}\bar{\mathbf{w}}||^2 - \lambda^{-2}\bar{\mathbf{w}}^\top\bar{\mathbf{w}}$.

It is very important to note that Eq. 6 and Eq. 11 are identical, with only the order of matrix multiplication changed, resulting in different complexity and model interpretation.

Random Fourier features (RFF) (Rahimi and Recht, 2007) can be used to approximate infinite stationary kernels with finite features based on Bochner's theorem:

Theorem 2.1 (Bochner's theorem, e.g. (Rudin, 2017))**.** *A continuous stationary kernel $k(\mathbf{x}, \mathbf{x}') = \tilde{k}(\mathbf{x} - \mathbf{x}') = \tilde{k}(\boldsymbol{\delta})$ on $\mathbb{R}^p$ is positive definite if and only if $\tilde{k}(\boldsymbol{\delta})$ is the Fourier transform of a non-negative measure.*

Above, $\boldsymbol{\delta} = \mathbf{x} - \mathbf{x}'$ and $\tilde{k}$ is the kernel k redefined in terms of one argument that is the difference of the inputs $\mathbf{x} - \mathbf{x}'$. If the kernel is real-valued and properly scaled, its inverse Fourier transform $p(\boldsymbol{\omega})$ is also real-valued and is a proper probability

distribution. Then a valid feature mapping is $\mathbf{z}_{\boldsymbol{\omega}}(\mathbf{x}) = (\cos(\boldsymbol{\omega}^\top \mathbf{x}), \sin(\boldsymbol{\omega}^\top \mathbf{x}))$, since

$$\tilde{k}(\boldsymbol{\delta}) = \int_{\mathbb{R}^p} p(\boldsymbol{\omega}) \cos(\boldsymbol{\omega}^\top \boldsymbol{\delta}) d\boldsymbol{\omega} = \mathbb{E}_{\boldsymbol{\omega}}[\cos(\boldsymbol{\omega}^\top \boldsymbol{\delta})] = \mathbb{E}_{\boldsymbol{\omega}}[\mathbf{z}_{\boldsymbol{\omega}}(\mathbf{x})^\top \mathbf{z}_{\boldsymbol{\omega}}(\mathbf{x}')]$$

The true expectation is then approximated by the empirical mean of $\mathbf{z}_{\boldsymbol{\omega}}(\mathbf{x})^\top \mathbf{z}_{\boldsymbol{\omega}}(\mathbf{x}')$ with multiple samples from $p(\boldsymbol{\omega})$, which enables recovering an infinite kernel with a finite set of features and working directly in the feature space.

2.2 Federated Learning v.s. Distributed Learning

In federated learning (FL), there is always a collection of clients $c \in S$ that wish to collaborate, and each client holds their own data $D_c = (\mathbf{X}_c, \mathbf{y}_c)$ locally. We seek to train some machine learning model $\mathcal{M}_\theta$ on these client data, with parameters θ. Conventionally, we would centralize all the data $D = \bigcup_{c \in S} D_c$, and learn the model globally. This simple approach becomes infeasible if the clients cannot directly share their data due to privacy concerns. FL is designed to specifically tackle this problem. In typical scenarios, a trusted central server is allowed to receive and send perturbed matrices that only contain limited information about raw client data such as model parameters.

Although FL seems similar to distributed learning (DL) where tasks are also distributed to client workers to exploit extra computational resources, FL focuses on settings that are very different from typical DL settings and thus faces unique challenges that DL algorithms cannot handle. First, DL algorithms are usually deployed on clusters, where communication between nodes are fast and reliable, thus algorithms with huge message throughput, like aggregating local gradients at each iterative step, are practical and feasible. However, in the *cross-device* setting of FL, client nodes are often millions of mobile devices that are not reliable and possess only limited computational resources. In this case, it is crucial to reduce the communication cost, so FL algorithms are often expected to aggregate local models or information that have evolved for more than one step or epoch of gradient descents from the previous global consensus, and a line of works (Konečnỳ et al., 2016; Sattler et al., 2019; Reisizadeh et al., 2020) are dedicated in improving the communication efficiency. This nature of clients further leads to data and system heterogeneity, since the server often, if not always, has no control over how data or computation resource is distributed. More importantly, FL algorithms focus on reducing privacy risks of clients, while it is not a concern at all of DL algorithms. Hence, many DL algorithms send risky messages with which other entities can easily retrieve the raw data of clients. A review with more details can be found in Appendix A.

2.3 Related work

Distributed and Federated GPs

Closely related to our work is the literature on distributed and federated GPs. Distributed GPs (Deisenroth and Ng, 2015; Yin et al., 2020) were initially proposed to improve scalability by partitioning the data into several machines since GPs that

operate in the dual space scale cubically with the amount of data in the worst case. The product-of-experts framework has emerged as a popular technique to aggregate local GPs, including generalized product of experts (Cao and Fleet, 2014) and robust Bayesian committee machines (Deisenroth and Ng, 2015). Distributed optimization of hyperparameters in GPs has also been explored using ADMM (Xie et al., 2019), statistical parameter matching (Yurochkin et al., 2019), or variational inference (Gal et al., 2014; Hoang et al., 2016). While those techniques do not ensure data privacy, recent work about federated GPs reduce privacy risks while training in a distributed way. This includes pFedGP (Achituve et al., 2021)and DEC-apx-GP (Kontoudis and Stilwell, 2023) which optimize the hyperparameters of a global kernel using either FedAvg or decentralized proximal ADMM. DEC-gapx-GP (Kontoudis and Stilwell, 2024) is an extension of DEC-apx-GP, which shares an additional dataset with all clients as extra training data. Note this violates the constraints of FL, in the absence of a public dataset, since the additional dataset must consist of client's local data. These federated GP algorithms share only the hyperparameters of kernels, but learn local GP posteriors that are never shared. In another line of work, GPs have also been used to estimate correlations between clients in FL in order to actively select independent clients for aggregation (Tang et al., 2021). Our work differs from previous distributed and federated GPs by learning a global federated GP with better generalization performance.

Deep kernel learning and GP approximations

There have been many works that committed to increasing the model capacity of GPs by incorporating deep neural networks (DNNs). Works of Hinton and Salakhutdinov (2007); Calandra et al. (2016) either pretrain a deep belief network or directly train it with a GP to extract first-step features before sending the data into the GP with conventional kernels. Wilson et al. (2016b,a) build their algorithm on top of Wilson and Nickisch (2015); Wilson et al. (2015); Titsias (2009); Hensman et al. (2013); Nickson et al. (2015), extending this idea with approximations for scalability and stochastic variational inference for classification tasks. Then Tran et al. (2019) studies the variance estimations of deep kernel learning models and proposes to use Monte-Carlo Dropout (Gal and Ghahramani, 2016) for better calibration. Ober et al. (2021) proposes to use Bayesian Neural Networks instead of deterministic DNNs to prevent over-fitting. Besides, Garnelo et al. (2018) designed a special architecture that makes it possible for DNNs to simulate GP behaviors. Damianou and Lawrence (2013) form GPs into a deep architecture that corresponds to a deep belief network based on GP mappings. To make GPs practical, one popular method is the inducing point approximation, where the joint GP prior of training points and inducing points are approximated (Quinonero-Candela and Rasmussen, 2005). Variants includes SoD (Quinonero-Candela and Rasmussen, 2005), SoR (Smola and Bartlett, 2000), DTC (Seeger et al., 2003), FITC (Snelson and Ghahramani, 2005), and PITC (Schwaighofer and Tresp, 2002). Later, KISS-GP (Wilson and Nickisch, 2015) gave another interpretation that inducing point approximations are equivalent to global GP interpolation, and it can exploit Kronecker structures (Saatçi, 2012). Another approximation method is random features (Rahimi and Recht, 2007; Sinha and Duchi, 2016; Oliva et al.,

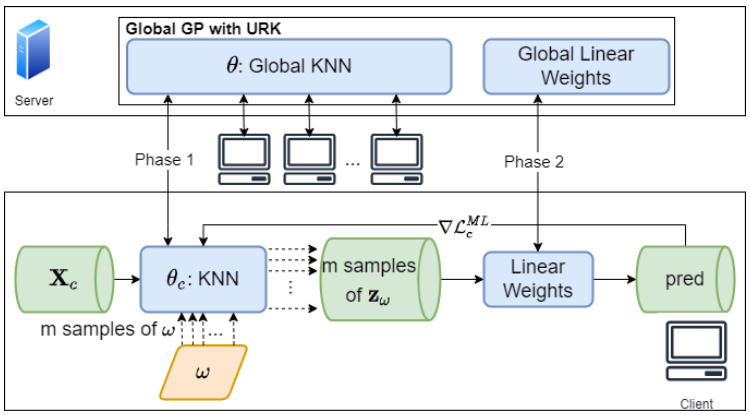

Fig. 1 FedBNR learns a global federated GP in two phases: kernel learning with federated optimization and last layer updating with exact Bayesian inference.

2016) that use randomized basis functions to approximate kernels. More information about scalable GPs can be found in this survey (Liu et al., 2020).

3 Federated Learning of Global GPs

Personalized federated learning (PFL) algorithms can learn fine-tuned local models for each client and can automatically handle preference distribution skew (Zhu et al., 2021) (i.e., $\Pr(\mathbf{y}|\mathbf{X})$ varies with clients). However, this comes at the cost of the generalization ability, especially when preference skew is not a concern and the clients want out-of-distribution generalization w.r.t. its training data. If a client is not capable of collecting enough data in the same range as their desired prediction, the accuracy of PFL local models could suffer. For example, suppose the posterior of a GP with a stationary kernel is conditioned on local training data $(\mathbf{x}_1, \cdots, \mathbf{x}_{n_c}) \subset \mathbb{B}_c$, an infinite compact set, and $(y_1, \cdots, y_{n_c})$. Due to non-i.i.d. clients, it is common that other clients c' have a different $\Pr(\mathbf{X})$ in the form that $\mathbb{B}_c \cap \mathbb{B}_{c'} \neq \mathbb{B}_{c'}$. That is, training data of client c' is considered out-of-range for client c. Suppose client c wants to generalize to the range of all clients' training data $\cup_{c\in S}\mathbb{B}_c$. The prediction of data points from other clients' range $\mathbf{x}' \notin \mathbb{B}_c$ will gradually become the constant zero prior as $\mathbf{x}'$ moves further from $\mathbb{B}_c$, regardless of what y' really is. Thus, this generalization error is unbounded. However, a global GP does exhibit a uniformly bounded error (Lederer et al., 2019), since its posterior is conditioned on all clients' data. It means all the clients, whether or not possessing training data, can benefit from others' information in FL.

With such motivation for learning global federated GPs, we derive our algorithm in the following sections. In Sec 3.1, we propose a random feature formulation URK, which is more flexible and compatible to deep learning as it can recover non-stationary kernels. It also provides more model capacity and thus better accuracy compared to plain deep learning without random features. Then, we detail our algorithm FedBNR in Sec 3.2, showing how a federated global posterior can be learned with URK. Finally, we extend FedBNR with DP in Sec 3.3, providing formal privacy guarantees.

3.1 GP under random feature maps

URK Definition

Recall that a kernel function can be rewritten as the inner product of some basis functions $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$. We propose the following kernel and random feature approximation.

Let $\boldsymbol{\omega} \in \mathbb{R}^{d'}$ be any random variable or vector, and $g : \mathbb{R}^{d'} \times \mathbb{R}^p \to \mathbb{R}^d$ be any function that extracts d features out of each input with random weights $\boldsymbol{\omega}$. We construct the random basis functions $\mathbf{z}$ as $\mathbf{z}_{\boldsymbol{\omega}}(\mathbf{x}) = g(\boldsymbol{\omega}, \mathbf{x})$. We define the true underlying kernel and its approximation, the unifying random kernel (URK) as:

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}_\omega[\mathbf{z}_\omega(\mathbf{x})^\top \mathbf{z}_\omega(\mathbf{x}')] \tag{12}$$

$$\approx \mathrm{URK}^m_{\omega,g}(\mathbf{x}, \mathbf{x}') := \left(\frac{\mathbf{s}^m_\omega(\mathbf{x})}{\sqrt{m-1}}\right)^\top \left(\frac{\mathbf{s}^m_\omega(\mathbf{x}')}{\sqrt{m-1}}\right) = \phi(\mathbf{x})^\top \phi(\mathbf{x}') \tag{13}$$

where $\mathbf{s}^m_{\boldsymbol{\omega}}(\mathbf{x})$ is the concatenation of m samples of $\mathbf{z}_{\boldsymbol{\omega}}(\mathbf{x})$. Equation 12 is the true underlying kernel we construct. Equation 13 is the Monte-Carlo approximation of Equation 12, where the samples are the random features that URK produces. Furthermore, note

$$\mathbb{E}_\omega[\mathbf{z}_\omega(\mathbf{x})^\top \mathbf{z}_\omega(\mathbf{x}')] = \mathrm{Tr}(\mathrm{Cov}_\omega(\mathbf{z}_\omega(\mathbf{x}), \mathbf{z}_\omega(\mathbf{x}'))) + \mathbb{E}_\omega[\mathbf{z}_\omega(\mathbf{x})]^\top \mathbb{E}_\omega[\mathbf{z}_\omega(\mathbf{x}')] \tag{14}$$

Equation 14 follows from the definition of covariance (i.e., $\mathrm{Cov}(A, B) = \mathbb{E}[AB] - \mathbb{E}[A]\mathbb{E}[B]$, where A, B are random variables). Tr refers to the trace, which is needed for random vectors. It is a decomposition of the true underlying kernel, which provides an intuitive understanding of the construction where URK mainly captures the covariance between different random basis functions. This aligns with the purpose of the kernel function.

Examples and Theoretical Guarantees

Since g is an arbitrary function, we can assign it any DNN with any architecture. We call it a kernel neural network (KNN) since its weights, θ, are essentially the kernel hyperparameters. It is trivial to map a deep kernel defined by $k(\phi_\theta(\mathbf{x}), \phi_\theta(\mathbf{x}'))$, where k is any conventional stationary kernel and ϕ_θ is any feature mapping function potentially represented by a deep neural network, into a URK construction:

$$\boldsymbol{\omega} \sim p(\boldsymbol{\omega}), g(\boldsymbol{\omega}, \mathbf{x}) = \left[\sin(\boldsymbol{\omega}^\top \phi_\theta(\mathbf{x})), \cos(\boldsymbol{\omega}^\top \phi_\theta(\mathbf{x}))\right]^\top \tag{15}$$

where $p(\boldsymbol{\omega})$ is the inverse Fourier transform of k. With m samples of ω, the error of approximating the true underlying kernel $k(\mathbf{x}, \mathbf{x}')$ drops exponentially:

Theorem 3.1. *$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^p, \forall p(\omega)$ being a proper distribution over $\mathbb{R}^{d'}$, and $g : \mathbb{R}^{d'} \times \mathbb{R}^p \to \mathbb{R}^d$ has the form of Eq. 15. With $m \in \mathbb{Z}_+$ samples of ω, $\forall t > 0$*

$$\Pr[|\, URK^m_{\omega,g}(\mathbf{x}, \mathbf{x}') - k(\mathbf{x}, \mathbf{x}')| \geq t] \leq 2\exp\left(-\frac{mt^2}{8}\right)$$

URK can reconstruct not only conventional stationary kernels with a RFF mapping, but also non-stationary kernels beyond RFF's capability. We give two example constructions and their corresponding convergence properties below. Full proofs are in Appendix D.

The first example recovers a non-stationary kernel that induces infinite basis functions, like the Gaussian kernel.

Proposition 3.2. *Let* $\omega \sim N(\mathbf{0}, \mathbf{I}), g(\omega, \mathbf{x}) = \exp(\omega^\top \mathbf{x})$*, then* $k(\mathbf{x}, \mathbf{x}') = \lim_{m\to\infty} URK^m_{\omega,g}(\mathbf{x}, \mathbf{x}')$ *is a non-stationary kernel with infinite features.*

The following theorem bounds the error of URK approximation for the construction in Prop. 3.2. Fenton-Wilkinson approximation (Fenton, 1960) is used to model the summation of log-normal distributions, and the error is bounded by the sum of two tail probabilities.

Theorem 3.3. *Let* ω *and* g *have the form of Prop. 3.2. Suppose* $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^p$ *satisfies* $||\mathbf{x}||_2^2 \le a, ||\mathbf{x}'||_2^2 \le a'$*. With* $m \in \mathbb{Z}_+$ *samples of* ω*,* $\forall t \in (0, \exp(a + a')/2))$*, by Fenton-Wilkinson approximation,*

$$\sigma_s := \log\left(\frac{\exp(a + a') - 1}{m} + 1\right), E := \exp((a + a')/2)$$

$$\Pr[|\, URK^m_{\omega,g}(\mathbf{x}, \mathbf{x}') - k(\mathbf{x}, \mathbf{x}')| \ge t] \le 1 - \boldsymbol{F}\left(\frac{2\log(t + E) - (a + a') + \sigma_s^2}{2\sigma_s}\right) + \boldsymbol{F}\left(\frac{2\log(E - t) - (a + a') + \sigma_s^2}{2\sigma_s}\right)$$

$\boldsymbol{F}$ *is the cumulative probability distribution function of the standard normal distribution.*

The second example recovers a popular non-stationary kernel, the polynomial kernel.

Proposition 3.4. *Let* $\tilde{c}, \tilde{n} \in \mathbb{R}$ *be hyperparameters of the polynomial kernel* $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + \tilde{c})^{\tilde{n}}$*. Define* $\tilde{\mathbf{p}} = [\frac{1}{2}, \frac{1}{2p}, \frac{1}{2p}, \cdots, \frac{1}{2p}]^\top \in \mathbb{R}^{p+1}$*,* $\omega \sim Multi(\tilde{n}, \tilde{\mathbf{p}})$*, the multinomial distribution. Define* $\tilde{\mathbf{x}} = [\sqrt{2\tilde{c}}, \sqrt{2p}\mathbf{x}^\top]^\top \in \mathbb{R}^{p+1}$*,* $g(\omega, \mathbf{x}) = \exp(\omega^\top \log \tilde{\mathbf{x}})$*, then* $k(\mathbf{x}, \mathbf{x}') = \lim_{m\to\infty} URK^m_{\omega,g}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + \tilde{c})^{\tilde{n}}$.

The following theorem bounds the error of URK approximation for the construction in Prop. 3.4, with Hoeffding's inequality (Hoeffding, 1994) similarly to Thm. 3.1.

Theorem 3.5. *Let* ω *and* g *have the form of Prop. 3.4. Suppose entries of* $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^p_+$ *are bounded by* $[a, b]$*. With* $m \in \mathbb{Z}_+$ *samples of* ω*,* $\forall t > 0$*,*

$$v := \frac{max\{\sqrt{2\tilde{c}}, \sqrt{2p}b\}}{min\{\sqrt{2\tilde{c}}, \sqrt{2p}a\}}$$

$$\Pr[|\, URK^m_{\omega,g}(\mathbf{x}, \mathbf{x}') - k(\mathbf{x}, \mathbf{x}')| \ge t] \le 2\exp\left(-\frac{2mt^2}{v^{4\tilde{n}}}\right)$$

Without Randomness

Note the deep features themselves already allow a valid BLR formulation that is equivalent to a linear kernel. However, it is restricted to finite basis functions as the

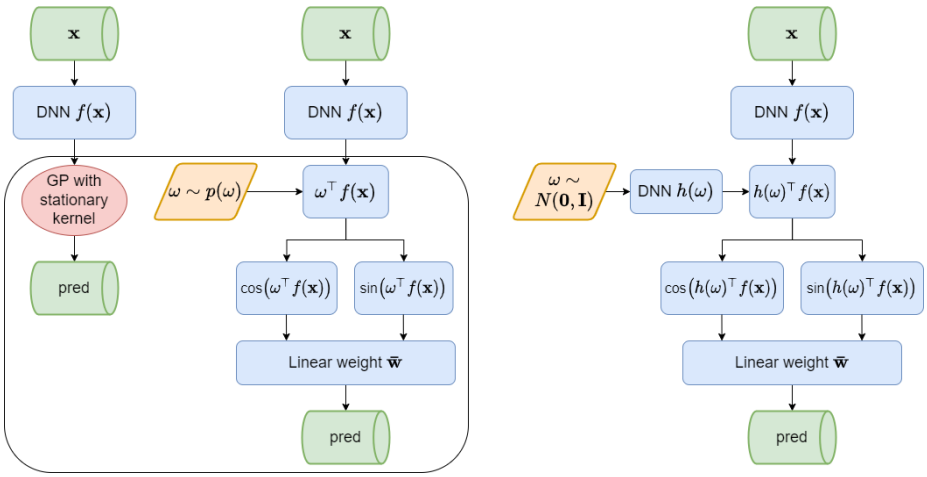

Fig. 2 From left to right: a standard deep kernel learning algorithm with conventional stationary kernel GPs; the corresponding URK architecture; a more general architecture enabled by URK for convenient latent stationary kernel learning.

same size of the second last layer and has the implicit assumption that the target is linear in the features. As the most basic kernel function, the linear kernel does not provide additional model capacity to the deep neural network. Previously, deep kernel learning algorithms (Wilson et al., 2016a,b) apply the Gaussian kernel, which induces infinite basis functions, to deep features and show they effectively improve the model accuracy compared to using deep features only. The rationale is straightforward: adding extra basis functions to a model never worsens its training accuracy, and GPs are resilient to over-fitting. Adding randomness with the URK construction follows the same idea. It further extends the model capacity of the KNN by allowing an effective approximation to infinite kernels. In Appendix C.1, we report an extra ablation study that shows empirically the URK approximation indeed improves the accuracy in most cases compared to deep features only.

Learn the Distribution

URK is meant to be a generic definition that enables any architecture g and any distribution for ω. Whether a specific instantiation is preferable is highly problem dependent. Nevertheless, we present an architecture that utilizes the flexibility of deep learning and let the model learn automatically a distribution for ω that best fits its data. The architecture is derived in alignment with deep kernel learning proposed by Wilson et al. (2016b). As illustrated in the left half of Fig. 2, URK can recover any DNN combined with a conventional stationary kernel, using the architecture mentioned. $p(\omega)$ of popular stationary kernels are reported in RFF (Rahimi and Recht, 2007). For example, the Gaussian kernel simply corresponds to $\omega \sim \mathcal{N}(0, \mathbf{I})$, the standard normal distribution. Furthermore, we can exploit the flexibility of URK and define a distribution shifter h that transforms ω. We can start from a standard normal distribution, which is very easy to sample from, and send the samples through h to simulate a much more complex distribution with minimal computational resources

required. We choose h such that the identity function is in its hypothesis set, which can be achieved by adding (potentially linearly transformed) ω to the output of any architecture, similarly to the residual network (He et al., 2016). Then, we can learn a distribution for ω that is at least as good as the Gaussian kernel in deep kernel learning.

3.2 FedBNR: federated learning of global GPs with URK

With URK, we can now approximate infinite deep kernels with finite random features and work in the feature space. The training procedure of our proposed algorithm FedBNR can be divided into two phases, as illustrated in Fig. 1. In the first phase, we train the KNN by optimization methods. In the second phase, we calculate the weights of the last linear layer that maps the random features to the output space, by a closed-form formula inferred from Equations 11. The pseudocode of FedBNR is summarized in Algorithm 1.

In phase 1, we follow a standard training procedure under the FL framework. We assume there is a central server holding the shared global KNN weights θ and global hyperparameters σ, λ that denote the noise level and the prior variance respectively. We assume there is a set of clients $c \in S$ holding local KNN weights θ_c, local hyperparameters σ_c, λ_c, local inputs $\mathbf{X}_c$, and local targets $\mathbf{y}_c$. We use $\phi_\theta(\omega, \mathbf{x})$ to denote the resulting features of $\mathbf{x}$ with a specific set of random samples ω with respect to θ. ω is omitted for simplicity when it is not fixed. In the beginning of each aggregation round, the server sends a copy of aggregated or initialized θ, σ, λ to all the clients. All the clients $c \in S$ first update their local model for a fixed number of iterations. Then they send θ_c, σ_c, and λ_c back to the server for aggregation and start another aggregation round. The local loss function is the negative local log marginal likelihood:

$$\begin{aligned}\mathcal{L}_c^{ML} =& -\log \Pr_{\text{BLR}}(\mathbf{y}_c|\phi_{\theta_c}(\mathbf{X}_c); \sigma_c, \lambda_c) \\ =& n_c \log \sigma_c^2 + \log(|\lambda_c^2 \mathbf{I}||\mathbf{A}_c|)||\mathbf{y}_c - \mathbf{\Phi}_c \bar{\mathbf{w}}_c||^2 + \lambda^{-2} \bar{\mathbf{w}}_\mathbf{c}^\top \bar{\mathbf{w}}_\mathbf{c}.\end{aligned} \tag{16}$$

One commonly used method for aggregation is the FedAvg (McMahan et al., 2017) heuristic, where the new global model parameters are assigned the average of all client model parameters $\theta = \sum_{c \in S} \theta_c / |S|$. However, multiple works (Karimireddy et al., 2020; Shoham et al., 2019; Mohri et al., 2019) have pointed out the quality and the convergence rate of this heuristic can suffer from non-i.i.d. clients. To account for this, we propose to adapt kernel knowledge distillation (He and Ozay, 2022) to aggregate the KNNs. We assume the server holds a relatively small dataset $\mathbf{X}_{kd}, \mathbf{y}_{kd}$ and tries to minimize the following knowledge distillation loss with respect to this dataset:

$$\mathcal{L}^{KD} = \mathcal{L}_{kd}^{ML} + \alpha * \text{MSE}(\phi_\theta(\mathbf{X}_{kd})^\top \phi_\theta(\mathbf{X}_{kd}), \sum_{c \in S} \phi_{\theta_c}(\mathbf{X}_{kd})^\top \phi_{\theta_c}(\mathbf{X}_{kd}) / |S|). \tag{17}$$

Here, $\mathcal{L}_{kd}^{ML}$ is the log marginal likelihood loss $\mathcal{L}^{ML}$ with respect to the knowledge distillation dataset $\mathbf{X}_{kd}, \mathbf{y}_{kd}$. α is a common hyperparameter in knowledge distillation methods to adjust the ratio between the log marginal likelihood loss and the mean squared error (MSE) loss. The MSE loss forces the global kernel $\phi_\theta(\mathbf{X}_{kd})^\top \phi_\theta(\mathbf{X}_{kd})$ to

simulate the mean of all client kernels, namely $\sum_{c\in S}\phi_{\theta_c}(\mathbf{X}_{kd})^\top\phi_{\theta_c}(\mathbf{X}_{kd})/|S|$, which is akin to concatenating all the features of the client kernels. Ideally, if the global kernel successfully learns to do so, it should not perform worse on any of the clients, while the FedAvg heuristic has no similar guarantees.

In phase 2, we fix the kernel hyperparameters and learn $\bar{\mathbf{w}}$ in an exact way as if all client data is centralized. First, note that we can decompose $\bar{\mathbf{w}}$ as follows:

$$\bar{\mathbf{w}} = \sigma^{-2}(\sigma^{-2}\mathbf{\Phi}\mathbf{\Phi}^\top + \lambda^{-2}\mathbf{I})^{-1}\mathbf{\Phi}\mathbf{y} \tag{18}$$

$$= \sigma^{-2}(\sigma^{-2}\sum_{c\in S}\mathbf{\Phi}_c\mathbf{\Phi}_c^\top + \lambda^{-2}\mathbf{I})^{-1}\sum_{c\in S}(\mathbf{\Phi}_c\mathbf{y}_c) \tag{19}$$

$\mathbf{\Phi}_c = \phi_\theta(\mathbf{X}_c)$ denotes the random features of local inputs extracted by the global KNN. The server first broadcasts the global model to all the clients and asks them to return the scatter matrices $\mathbf{\Phi}_c\mathbf{\Phi}_c^\top$ and the vectors $\mathbf{\Phi}_c\mathbf{y}_c$. Then, the server simply takes the summation of client messages, calculates $\bar{\mathbf{w}}$ according to Equation 19, and broadcasts the whole model again to all the clients. Note that phase 2 only needs to be done once after kernel learning to achieve the global model. There is no need to iterate between the two phases.

In phase 1, each local update costs $O(n_c(md)^2+(md)^3)$ time to compute and inverse $\mathbf{A}_c$ and $O((md)^2)$ space to store $\mathbf{A}_c$. This is equivalent to the complexity of BLR models with random feature approximations. It is more scalable than working with the kernel $\mathbf{K}$ directly, which costs $O(n_c^3)$ time and $O(n_c^2)$ space when $md \ll n_c$. Each client shares a message of size $|\boldsymbol{\theta}|$ with the server. FedAvg aggregation then costs $O(|\boldsymbol{\theta}|)$ time and $O(|\boldsymbol{\theta}|)$ space, while KD aggregation costs $O(n_{kd}(md)^2 + (md)^3 + n_{kd}^2md)$ time and $O((md)^2 + n_{kd}^2)$ space, where n_{kd} is the size of the knowledge distillation dataset. In phase 2, each client calculates the scatter matrices and the vectors, costing $O(n_c(md)^2)$ time and $O((md)^2)$ space. Each client shares a message of size $(md)^2$ with the server. Finally, learning the global posterior costs $O((md)^3)$ time and $O((md)^2)$ space, similarly to phase 1. Note that if n_{id} inducing points are used to approximate the full kernel (e.g. in pFedGP), $O(n_{id}^3)$ time and $O(n_{id}^2)$ space is needed similarly.

3.3 Enhance protection with DP

We claim that FedBNR protects the privacy of clients at least as well as FedAvg and other FL algorithms that send client models to the server. In phase 1, the aggregation only requires client model parameters. In phase 2, we send two messages outside each client: $\mathbf{\Phi}_c\mathbf{\Phi}_c^\top$ and $\mathbf{\Phi}_c\mathbf{y}_c$. These matrices and vectors (of size $m \times m$ and m) are also highly compressed summaries of the raw data, which prevents immediate privacy invasions as discussed in Appendix A. We note however that FL with compressed messages does not prevent privacy attacks (Boenisch et al., 2021; Mothukuri et al., 2021), it simply makes it harder to infer the underlying data. To provide formal privacy guarantees, federated learning needs to be combined with formal mechanisms such as differential privacy (DP) (Wei et al., 2020; Triastcyn and Faltings, 2019) or secure multi-party computation (MPC) (Truex et al., 2019; Byrd and Polychroniadou, 2020; Li et al., 2020).

Algorithm 1 Federated Bayesian Neural Regression
(θ: the global KNN, σ: noise level, λ: prior covariance of the linear layer, ω: a set of random numbers, ζ: local learning step, ζ': knowledge distillation step)

Phase 1: Kernel Learning
Initialize $\theta \leftarrow \theta^0, \sigma \leftarrow \sigma^0, \lambda \leftarrow \lambda^0$
for each aggregation round $t \leftarrow 0, 1, 2, \cdots$ **do**
  for each client $c \in S$ **do**
    $\theta_c^t, \sigma_c^t, \lambda_c^t \leftarrow \theta^t, \sigma^t, \lambda^t$
    for each local update round **do**
      $\mathbf{w}_c = (\sigma_c^t)^{-2}\mathbf{A}_c^{-1}\mathbf{\Phi}_c\mathbf{y}_c$, $\theta_c^t, \sigma_c^t, \lambda_c^t$ `-=` $\zeta\nabla\mathcal{L}_c^{ML}$ (Eq. 16)
    end for
  end for
  if FedAvg **then**
    $\theta^{t+1}, \sigma^{t+1}, \lambda^{t+1} \leftarrow \bar{\theta_S^t}, \bar{\lambda_S^t}, \bar{\lambda_S^t}$
  else if Knowledge Distillation **then**
    $\theta^{t+1}, \sigma^{t+1}, \lambda^{t+1} \leftarrow \theta^t, \sigma^t, \lambda^t$
    for each knowledge distillation round **do**
      $\theta^{t+1}, \sigma^{t+1}, \lambda^{t+1}$ `-=` $\zeta'\nabla\mathcal{L}^{KD}$ (Eq. 17)
    end for
  end if
end for
Phase 2: Update the Global Linear Layer
Server: send $\theta, \sigma, \lambda, \omega$ to all clients
for each **Client** $c \in S$ **do**
  compute the random features $\mathbf{\Phi}_c \leftarrow \phi_\theta(\omega, \mathbf{X}_c)$, send $\mathbf{\Phi}_c\mathbf{\Phi}_c^\top, \mathbf{\Phi}_c\mathbf{y}_c$ to the server
end for
Server: $\mathbf{A}^{-1} \leftarrow (\sigma^{-2}\sum_c(\mathbf{\Phi}_c\mathbf{\Phi}_c^\top)+\lambda^{-2}\mathbf{I})^{-1}$, $\bar{\mathbf{w}} \leftarrow \sigma^{-2}\mathbf{A}^{-1}\sum_c(\mathbf{\Phi}_c\mathbf{y}_c)$, broadcast $\bar{\mathbf{w}}$

Definition 1. A mechanism M satisfies (ϵ, δ)-DP if for any two datasets D, D' that differ by only one record (i.e. $|(D - D') \cup (D' - D)| = 1$, noted as $D \sim D'$), and for any possible output $O \in Range(M)$,

$$\Pr_{O\sim M(D)}\left[\log(\frac{\Pr[M(D) = O]}{\Pr[M(D') = O]}) > \epsilon\right] < \delta.$$

The idea of (ϵ, δ)-DP is to add enough noise to the shared information so that with high probability ($0 \leq \delta < 1$), the distribution of the shared message does not change much when the raw dataset is modified by one record (controlled by $\epsilon > 0$). Thus, one cannot effectively tell if any record is included in the dataset. (ϵ, δ)-DP models the information leakage of the raw dataset with the inner log ratio of Def. 1, and guarantees that it is bounded by ϵ, the privacy budget, with probability δ. Empirically, people set $\epsilon \leq 1$ for strong privacy protection, while $\epsilon \geq 10$ is deemed weak. We will use the following mechanism:

Theorem 3.6 (e.g. (Kerkouche et al., 2021)). *A Gaussian Mechanism $\mathcal{GM}$ is (ϵ, δ)-DP with information revealed f: $\mathcal{GM}(D) = f(D) + \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, where $\sigma = L_2(f)\sqrt{2\ln(1.25/\delta)}/\epsilon$.*

$L_2(f)$ calculates the L_2 sensitivity of function f as follows.

$$L_2(f) = \max_{D,D'} ||f(D) - f(D')||_2, D \sim D'. \tag{20}$$

We now extend FedBNR to be differentially private with the Gaussian mechanism. In phase 1, the L_2 sensitivity of the kernel hyperparameters is the L_2 gradient clipping constant times the number of local update rounds times the learning rate ζ. In phase 2, we send $[\mathbf{\Phi}_c\mathbf{\Phi}_c^\top, \mathbf{\Phi}_c\mathbf{y}_c, \mathbf{y}_c^\top\mathbf{y}_c]$ instead and its L_2 sensitivity is $\sqrt{32 + 16/m + 8b^2 + b^4}$, where $b := max(\mathbf{y}) - min(\mathbf{y})$, assuming the URK construction is $g(\boldsymbol{\omega}, \mathbf{x}) = [\sin(\boldsymbol{\omega}^\top\phi_{\text{NN}}(\mathbf{x})), \cos(\boldsymbol{\omega}^\top\phi_{\text{NN}}(\mathbf{x}))]^\top$. See details in Appendix A. Then, notice the following matrix is PSD without DP:

$$\begin{bmatrix}\mathbf{\Phi}_c \\ \mathbf{y}_c^\top\end{bmatrix}\begin{bmatrix}\mathbf{\Phi}_c^\top & \mathbf{y}_c\end{bmatrix} = \begin{bmatrix}\mathbf{\Phi}_c\mathbf{\Phi}_c^\top & \mathbf{\Phi}_c\mathbf{y}_c \\ (\mathbf{\Phi}_c\mathbf{y}_c)^\top & \mathbf{y}_c^\top\mathbf{y}_c\end{bmatrix}. \tag{21}$$

After adding Gaussian noise, we project the matrix back to the nearest PSD matrix by setting its negative eigenvalues to 0. Then we perform regular BLR with projected $\mathbf{\Phi}_c\mathbf{\Phi}_c^\top$ and $\mathbf{\Phi}_c\mathbf{y}_c$.

4 Experiments

4.1 Synthetic experiments

We designed the following synthetic experiments to **show local GP models cannot handle generalization or new clients**. We first decide a true underlying function, sample 200 points uniformly from the range $[-5, 5]$, and add Gaussian noise with $\sigma = 0.5$. We then learn the kernel hyperparameters in a centralized fashion to eliminate any impact from imperfect kernels later on. We assign the first 100 points in range $[-5, 0]$ to client 1, and the rest to client 2. We train pFedGP and FedBNR with the learned kernel hyperparameters fixed and query for prediction in range $[-5, 5]$. The top left graph of Figure 3 shows the result of pFedGP, and the top right one shows the result of FedBNR. The blue curve shows the true underlying function, and the green curve shows the predictions. The light red area is a 95% confidence interval based on variance. The purple dots are the data leveraged by each model for the last layer. Since both algorithms learn hyperparameters centrally, the only difference lies in whether they leverage data of both clients for the predictive model. As shown in the pFedGP graph, the personalized model of client 1 only sees its own data, so it does not generalize well to the range of client 2, while the global model learned by FedBNR leverages all the data and generalizes to the full range.

Next, we introduce two new clients into the system. The first client holds data uniformly sampled from the range $[-5, 5]$, and the second from the range $[5, 15]$. We again fix the kernel hyperparameters learned centrally and train pFedGP, FedLoc, and FedBNR on these new clients separately. For testing, we still query the range

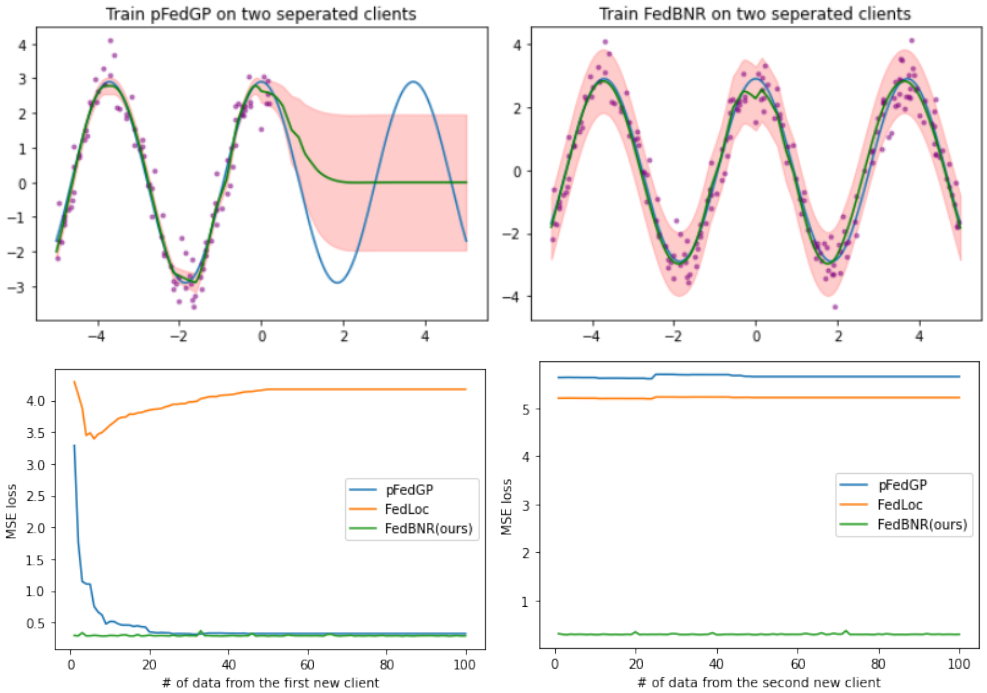

Fig. 3 Top graphs: (i) left: prediction of pFedGP trained on two non-overlapping clients; (ii) right: global prediction of FedBNR. Blue curve: underlying truth; green curve: point estimation; red range: 95% confidence interval; purple dots: training data leveraged by the model. Bottom graphs: MSE of predictions tested on the same range v.s. number of training data from a new client; (i) left: client with training range [-5, 5]; (ii) right: client with training range [5, 15]

$[-5, 5]$. As shown in the bottom graphs of Figure 3. The quality of prediction of the pFedGP model is massively impacted when there are few points in the first case, and for FedBNR the loss remains approximately a straight line. For FedLoc, the loss is also impacted due to zero-out effects of non-overlapping client models. Even worse, when training data and testing data are not in the same range for the second new client, the MSE loss of both pFedGP and FedLoc never drops back to the level of previous clients.

4.2 Real world regression tasks

To **show our algorithm achieves better generalization performance in terms of both point and uncertainty estimation**, we conducted comprehensive experiments on 5 real world regression tasks under 10 cases, including Skillcraft (Thompson et al., 2013), SML (Zamora-Martinez et al., 2014), Parkinsons (Tsanas et al., 2009), Bike (VE and Cho, 2020), and CCPP (Tüfekci, 2014). We note while a common challenge for GPs is to scale to large datasets, in FL, the challenge is the opposite: how to combine models trained on small datasets. When the datasets of each client become sufficiently large, then there is no need to combine models nor do FL. Hence, the

Table 1 Real world regression datasets, RMSE reported for the generalization setting. $\Uparrow$ and $\uparrow$ denote significantly worse results with $p < 0.01$ and $p < 0.05$ respectively; $\Downarrow$ and $\downarrow$ denote significantly better results similarly.

	Skillcraft		SML		Parkinsons		Bike		CCPP	
train/test size	2670/334		3309/414		4700/587		7008/876		7654/957	
corr-coef	-0.660		0.783		0.410		0.539		-0.948	
#clients	10	100	10	100	10	100	10	100	10	100
Central GP	0.95		0.21		3.58		0.37		4.02	
local+local	$1.26^{\Uparrow}$	$1.48^{\Uparrow}$	$1.49^{\Uparrow}$	$2.32^{\Uparrow}$	$10.9^{\Uparrow}$	$10.6^{\Uparrow}$	$0.79^{\Uparrow}$	$0.92^{\Uparrow}$	$14.6^{\Uparrow}$	$19.3^{\Uparrow}$
local+global	$1.08^{\Uparrow}$	$1.22^{\Uparrow}$	$1.00^{\Uparrow}$	$1.65^{\Uparrow}$	$6.42^{\Uparrow}$	$7.42^{\Uparrow}$	$0.59^{\Uparrow}$	$0.73^{\Uparrow}$	$5.62^{\Uparrow}$	$7.03^{\Uparrow}$
avg+local	$1.05^{\Uparrow}$	$1.06^{\Uparrow}$	$0.61^{\Uparrow}$	$0.81^{\Uparrow}$	$9.84^{\Uparrow}$	$8.80^{\Uparrow}$	$0.45^{\Uparrow}$	$0.54^{\Uparrow}$	$8.32^{\Uparrow}$	$13.4^{\Uparrow}$
kd+local	$1.04^{\Uparrow}$	$1.28^{\Uparrow}$	$0.86^{\Uparrow}$	$1.34^{\Uparrow}$	$6.15^{\Uparrow}$	$6.97^{\Uparrow}$	$0.51^{\Uparrow}$	$0.61^{\Uparrow}$	$10.0^{\Uparrow}$	$17.1^{\Uparrow}$
FedAvg	$1.00^{\uparrow}$	$1.03^{\Uparrow}$	$0.36^{\Uparrow}$	$0.71^{\Uparrow}$	$6.83^{\Uparrow}$	$7.38^{\Uparrow}$	$\mathbf{0.38}^{\downarrow}$	0.42	4.45	4.44
FedProx	0.98	$1.05^{\Uparrow}$	$0.34^{\uparrow}$	$0.62^{\Uparrow}$	$6.32^{\Uparrow}$	$7.38^{\Uparrow}$	0.39	0.42	4.43	4.47
pFedGP	$0.99^{\uparrow}$	$1.15^{\Uparrow}$	$0.75^{\Uparrow}$	$1.34^{\Uparrow}$	$9.36^{\Uparrow}$	$8.91^{\Uparrow}$	$0.45^{\Uparrow}$	$0.46^{\Uparrow}$	$14.2^{\Uparrow}$	$18.2^{\Uparrow}$
FedLoc	$1.08^{\Uparrow}$	$4.15^{\Uparrow}$	$2.83^{\Uparrow}$	$5.43^{\Uparrow}$	$8.40^{\Uparrow}$	$11.5^{\Uparrow}$	$0.64^{\Uparrow}$	$0.75^{\Uparrow}$	$20.6^{\Uparrow}$	$44.1^{\Uparrow}$
DEC-apx	$1.10^{\Uparrow}$	$4.34^{\Uparrow}$	$1.90^{\Uparrow}$	$4.93^{\Uparrow}$	$8.59^{\Uparrow}$	$9.21^{\Uparrow}$	$0.76^{\Uparrow}$	$0.88^{\Uparrow}$	$7.64^{\Uparrow}$	$11.06^{\Uparrow}$
DEC-gapx	$\mathbf{0.94}^{\Downarrow}$	$4.31^{\Uparrow}$	$1.41^{\Uparrow}$	$3.57^{\Uparrow}$	$8.07^{\Uparrow}$	$8.26^{\Uparrow}$	$0.55^{\Uparrow}$	$0.75^{\Uparrow}$	$6.26^{\Uparrow}$	$9.12^{\Uparrow}$
ours										
FedBNR	0.98	**0.97**	**0.25**	**0.44**	**3.10**	5.42	0.39	**0.42**	4.40	4.51
FedBNR-KD	$0.96^{\Downarrow}$	$0.98^{\uparrow}$	$0.55^{\Uparrow}$	$0.55^{\Uparrow}$	$4.58^{\Uparrow}$	$\mathbf{4.67}^{\downarrow}$	$0.43^{\Uparrow}$	$0.48^{\Uparrow}$	**4.38**	$\mathbf{4.38}^{\downarrow}$

experiments focus on reducing the amount of data per client while increasing the number of clients. The scalability of FedBNR is trivially guaranteed by the use of random features. Results of two variants are reported: i) *FedBNR* that performs the FedAvg heuristic at aggregation; and ii) *FedBNR-KD* that performs the knowledge distillation method at aggregation. Two groups of baselines are compared to our method for RMSE error: i) ablation study that includes local+local, local+global, avg+local, and kd+global, in the format of kernel learning method + last linear weight learning method, where we remove the global aggregation of either Phase 1 or Phase 2 from our methods; ii) previous works that include (1) FedAvg (McMahan et al., 2017), a standard non-Bayesian FL algorithm that has a global model; (2) FedProx (Li et al., 2020), a non-Bayesian FL algorithm that adds a proximal loss to FedAvg to prevent client models from getting too far from the global model; (3) pFedGP (Achituve et al., 2021), a Bayesian PFL method that learns a local GP with a shared deep kernel for each client using FedAvg; (4) FedLoc (Yin et al., 2020), a Bayesian FL method that directly applies distributed GP methods (Xie et al., 2019) without deep kernel learning; (5) DEC-apx-GP (Kontoudis and Stilwell, 2023), a decentralized Bayesian PFL method that learns a local GP with a shared separable squared exponential kernel for each client using proximal ADMM; and (6) DEC-gapx-GP (Kontoudis and Stilwell, 2024), an extension of DEC-apx-GP that shares an additional dataset with all clients as extra training data. We also compare to pFedGP, FedLoc, DEC-apx, and

Table 2 ECE reported for the generalization setting. ⇑, ↑, ⇓, and ↓ denote significantly worse or better results similarly to Table 1.

	Skillcraft		SML		Parkinsons		Bike		CCPP	
#clients	10	100	10	100	10	100	10	100	10	100
Central GP	0.02		0.09		0.27		0.11		0.24	
pFedGP	$0.43^{\Uparrow}$	$0.38^{\Uparrow}$	$0.45^{\Uparrow}$	$0.45^{\Uparrow}$	$0.49^{\Uparrow}$	$0.48^{\Uparrow}$	$0.42^{\Uparrow}$	$0.41^{\Uparrow}$	$0.49^{\Uparrow}$	$0.49^{\Uparrow}$
FedLoc	$0.32^{\Uparrow}$	$0.44^{\Uparrow}$	$\mathbf{0.12}^{\Downarrow}$	$0.27^{\Uparrow}$	$0.32^{\Uparrow}$	$0.43^{\Uparrow}$	$0.26^{\Uparrow}$	$0.16^{\Uparrow}$	$\mathbf{0.23}^{\Downarrow}$	$0.50^{\Uparrow}$
DEC-apx	$0.50^{\Uparrow}$	$0.46^{\Uparrow}$	$0.50^{\Uparrow}$	$0.50^{\Uparrow}$	$0.50^{\Uparrow}$	$0.50^{\Uparrow}$	$0.19^{\Uparrow}$	$0.14^{\Uparrow}$	$0.42^{\Uparrow}$	$0.38^{\Uparrow}$
DEC-gapx	$0.50^{\Uparrow}$	$0.36^{\Uparrow}$	$0.50^{\Uparrow}$	$0.50^{\Uparrow}$	$0.50^{\Uparrow}$	$0.50^{\Uparrow}$	$0.28^{\Uparrow}$	$0.19^{\Uparrow}$	$0.47^{\Uparrow}$	$0.42^{\Uparrow}$
ours										
FedBNR	**0.05**	$0.20^{\Uparrow}$	$0.39^{\Uparrow}$	$0.37^{\Uparrow}$	$0.36^{\uparrow}$	$0.40^{\Uparrow}$	**0.07**	$0.04^{\Uparrow}$	$0.24^{\Downarrow}$	$\mathbf{0.20}^{\Downarrow}$
FedBNR-KD	**0.05**	**0.06**	0.20	**0.21**	**0.29**	**0.30**	0.08	**0.09**	0.30	0.31

DEC-gapx for calibration errors since they are Bayesian models that have a notion of confidence. Besides, we report results of a centralized GP equipped with URK as a casual reference of the testing error lower bound. We used fully connected neural networks to extract first-step features for FedAvg, FedProx, and pFedGP. We used Gaussian kernels for the GPs in pFedGP and FedLoc. For fairness, KNNs used by our methods have similar architectures to the combined kernel of pFedGP. FITC (Snelson and Ghahramani, 2005) approximations are implemented for pFedGP and FedLoc as described in pFedGP. We used 50 random samples for KNN and 50 inducing points for pFedGP and FedLoc. Further details are in Appendix B.

Each dataset is uniformly divided into training, testing, and validation sets globally with a ratio of 8:1:1. The training data is sorted by the feature that has the largest absolute correlation coefficient with the output, and divided into multiple chunks. Each client randomly takes two chunks so their data distributions are heterogeneous. The larger the absolute correlation coefficient is, the more significant the distribution skew is. Before training, we tune hyperparameters that cannot be learned by gradient descent with grid searching on the validation set. Specially, FedBNR-KD uses 80% of the validation set for knowledge distillation, and the rest 20% for validation. All datasets are ran for 50 local epochs times at most 100 aggregation rounds in full batches. Validation and testing errors are recorded for each aggregation round. For methods that contain only local models, these errors are defined as the mean error of all local models with respect to the testing/validation set. If the validation error has not improved for 5 rounds, the training process is terminated.

We report the average minimum testing RMSE for 10 random seeds for each case in Table 1, which assesses the point estimation of different models. We also measured the statistical significance of the results compared to FedBNR with one-tailed Wilcoxon signed-rank tests (Wilcoxon, 1992). We report expected calibration errors (ECE) in Table 2 and perform the Wilcoxon test compared to FedBNR-KD. ECE is a calibration metric that assesses the uncertainty estimation. Further metrics are included in Appendix C.

Table 3 RMSE (top half) and ECE (bottom half) reported for the personalization setting. ⇑ and ↑ denote significantly worse results with $p < 0.01$ and $p < 0.05$ respectively; ⇓ and ↓ denote significantly better results similarly.

	Skillcraft		SML		Parkinsons		Bike		CCPP	
#clients	10	100	10	100	10	100	10	100	10	100
pFedGP	1.00$^{\Downarrow}$	**1.08**$^{\Downarrow}$	0.31$^{\Uparrow}$	0.64	2.97$^{\Uparrow}$	3.66$^{\Uparrow}$	**0.47**	**0.44**$^{\Downarrow}$	4,72$^{\Uparrow}$	12.66$^{\Uparrow}$
pxADMM	0.99$^{\Downarrow}$	1.16$^{\Downarrow}$	2.11$^{\Uparrow}$	1.75$^{\Uparrow}$	6.99$^{\Uparrow}$	3.73$^{\Uparrow}$	0.56$^{\Uparrow}$	0.50$^{\Downarrow}$	10.21$^{\Uparrow}$	14.60$^{\Uparrow}$
DEC-apx	1.10	4.19$^{\Uparrow}$	1.59$^{\Uparrow}$	3.49 $^{\Uparrow}$	3.56$^{\Uparrow}$	3.32$^{\Uparrow}$	0.57$^{\Uparrow}$	0.71$^{\Uparrow}$	4.40$^{\Uparrow}$	4.92$^{\Uparrow}$
DEC-gapx	**0.96**$^{\Downarrow}$	2.91$^{\Uparrow}$	1.15$^{\Uparrow}$	3.30$^{\Uparrow}$	4.49$^{\Uparrow}$	5.83$^{\Uparrow}$	0.54$^{\Uparrow}$	0.64$^{\Uparrow}$	4.40$^{\Uparrow}$	5.74$^{\Uparrow}$
ours										
FedBNR	1.12	1.25	**0.25**	**0.62**	**1.39**	**1.75**	**0.47**	0.57	**3.89**	4.77
FedBNR-KD	1.12	1.22$^{\Downarrow}$	0.38$^{\Uparrow}$	0.66	1.78$^{\Uparrow}$	2.13$^{\Uparrow}$	0.49$^{\Uparrow}$	0.63$^{\Uparrow}$	3.95$^{\Uparrow}$	**4.74**

	Skillcraft		SML		Parkinsons		Bike		CCPP	
#clients	10	100	10	100	10	100	10	100	10	100
pFedGP	0.42$^{\Uparrow}$	0.35$^{\Uparrow}$	0.45$^{\Uparrow}$	0.44$^{\Uparrow}$	0.45$^{\Uparrow}$	0.44$^{\Uparrow}$	0.42$^{\Uparrow}$	0.39$^{\Uparrow}$	0.49$^{\Uparrow}$	0.49$^{\Uparrow}$
pxADMM	0.42$^{\Uparrow}$	0.34$^{\Uparrow}$	0.50$^{\Uparrow}$	0.45$^{\Uparrow}$	0.49$^{\Uparrow}$	0.44$^{\Uparrow}$	0.20$^{\Uparrow}$	0.21$^{\Uparrow}$	0.50$^{\Uparrow}$	0.49$^{\Uparrow}$
DEC-apx	0.50$^{\Uparrow}$	0.50$^{\Uparrow}$	0.50$^{\Uparrow}$	0.50$^{\Uparrow}$	0.50$^{\Uparrow}$	0.50$^{\Uparrow}$	0.27$^{\Uparrow}$	0.21$^{\Uparrow}$	0.27$^{\Uparrow}$	0.43$^{\Uparrow}$
DEC-gapx	0.50$^{\Uparrow}$	0.50$^{\Uparrow}$	0.50$^{\Uparrow}$	0.50$^{\Uparrow}$	0.45$^{\Uparrow}$	0.50$^{\Uparrow}$	0.28$^{\Uparrow}$	0.24$^{\Uparrow}$	**0.20**$^{\Downarrow}$	**0.12**$^{\Downarrow}$
ours										
FedBNR	**0.01**	0.04	0.24	0.28	**0.04**	**0.04**	**0.11**	0.08	0.25	0.33
FedBNR-KD	**0.01**	**0.03**$^{\Downarrow}$	**0.13**$^{\Downarrow}$	**0.17**$^{\Downarrow}$	0.06$^{\Uparrow}$	0.09$^{\Uparrow}$	**0.11**	**0.07**	0.29$^{\Uparrow}$	0.33

The results show: i) in terms of RMSE, our methods are statistically better than the other models in most of the cases, especially when the training set at each client is significantly smaller than the whole set; ii) compared with the ablation study methods, our methods always perform better, so the global aggregation at both phases is essential; iii) FedLoc without deep kernel has a much smaller model capacity; iv) FedBNR-KD only outperforms FedBNR in 40% of cases in terms of RMSE, which is probably due to the small size of data (8% of all) used for knowledge distillation. However, it is clearly more stable when client heterogeneity gets worse as the number of clients increases. It is also better calibrated in most cases.

4.3 Personalization

We conducted experiments with the personalization setting to **show that the global GP posterior benefits uncertainty estimation even in the personalization setting**. FedBNR can be naturally extended to the personalization setting by fine-tuning the global model. During fine-tuning, we minimize $\mathcal{L}_c^{FT} = ||\mathbf{y}_c - \phi_{\theta_c}(\mathbf{X}_c)\mathbf{w}_c||_2^2 + \lambda_c^{-2}||\mathbf{w}_c||_2^2$, the local regularized least square loss. Note the weights in the last linear layer are also updated by gradient descent. Varying the number of epochs of fine-tuning offers a controllable trade-off between generalization and personalization. Training datasets are distributed to clients as before for heterogeneity, but testing and validation sets are sampled with the same distribution as local training datasets, with a ratio of

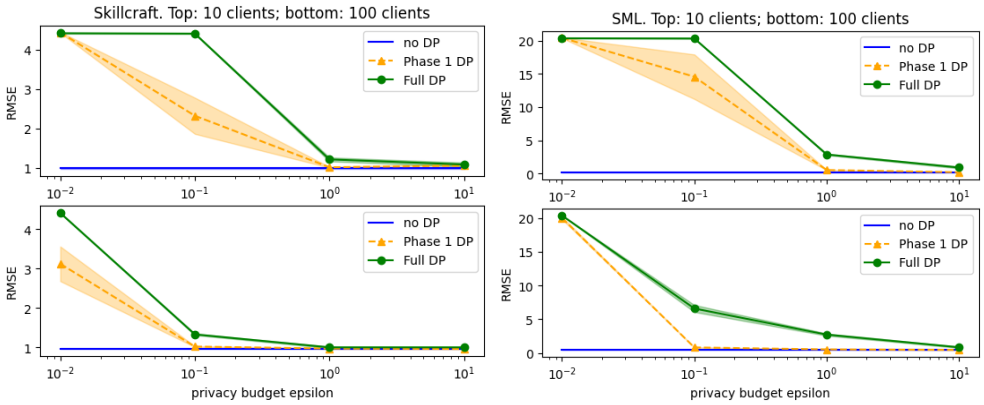

Fig. 4 FedBNR+DP results. Left: Skillcraft dataset; right: SML dataset. Phase 1 DP: add DP protection to θ_c only; full DP: add DP protection to all messages.

14:3:3. We report the RMSE and ECE in Table 3, where pxADMM (Xie et al., 2019) is a distributed GP hyperparameter learning algorithm used by FedLoc and we extend it with deep kernels. The results show that FedBNR achieves comparable accuracy and better calibration than other methods, indicating a global posterior helps calibrate local models.

4.4 Differential privacy

We conducted experiments for FedBNR with DP to **show the trade-off between model utility and the privacy budget** ϵ. Experiment details are similar to Section 4.2. The results of the Skillcraft and SML datasets are summarized in Figure 4. As the graph shows, the RMSE loss of fully differentially private FedBNR quickly drops back to the RMSE of vanilla FedLog, when $\epsilon \geq 0.1$. The DP noise affects the model less when there are more clients, since in Phase 1 the messages are averaged. Extra results are in Appendix C.

5 Conclusion

In this work, we proposed FedBNR, a novel algorithm that learns a global federated GP that provides better generalization and uncertainty estimation. FedBNR learns a deep kernel under feature mapping, and shares scatter matrices instead of direct features to achieve the exact global optimum of the last layer. It shows empirically statistically significant improvements in terms of point and uncertainty estimation compared to other federated GPs. One limitation of FedBNR is that the decomposition of global posterior learning with random features is restricted to regression tasks, and it is non-trivial to extend to the GP classification case where Laplacian approximation is needed.

Supplementary information. We provide public code in the supplementary file.

Acknowledgements. Resources used in preparing this research at the University of Waterloo were provided by Huawei Canada, the province of Ontario and the government of Canada through CIFAR and companies sponsoring the Vector Institute.

Declarations

Funding

The research leading to these results received funding from Huawei Canada under Grant Agreement #TC20210831001 and the AI Chair program of the Canadian Institute for Advanced Research (CIFAR).

Competing interests

Financial interests: The authors declare they have no financial interests.
Non-financial interests: Author Pascal Poupart is on the editorial board of the *Machine Learning* and receives no compensation as member of the editorial board.

Ethics approval and consent to participate

Not applicable.

Consent for publication

All authors consent to the submission of this manuscript to the *Machine Learning*.

Data availability

All datasets used in the experiments are cited and available for download online.

Materials availability

Not applicable.

Code availability

We provide public code in the supplementary file.

Author contribution

Haolin Yu, Kaiyang Guo, Mahdi Karami, Guojun Zhang, and Pascal Poupart contributed to the study conception and design. Material preparation, data collection and analysis were performed by Haolin Yu. The first draft of the manuscript was written by Haolin Yu and Pascal Poupart. All authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Table A.1 Privacy leakage of different GP algorithms. Risky message: messages that leak privacy but unavoidable for global prediction. Problematic term: terms that the risky messages try to compute. Leaked info: private information that can be recovered by adversaries with the risky messages. $\subset$ means "part of".

Technique	Risky message	Problematic term	Leaked info
$\mathbf{X}_c$ or $\mathbf{\Phi}_c$	—	—	$\mathbf{X}_c$, $\mathbf{y}_c$
Inducing points $\mathbf{U}$ or $\mathbf{U}_c$	—	—	$\subset \mathbf{X}_c$
$k(\mathbf{X}_c, \mathbf{U})$	—	—	$\mathbf{y}_c$
FedAvg (2017)	N/A	N/A	N/A
pFedGP (2021)	N/A	N/A	N/A
FedBNR (ours)	N/A	N/A	N/A
Centralized GP	$\mathbf{X}_c/\mathbf{\Phi}_c$	$\mathbf{K}$, $k(\mathbf{x}_\star, \mathbf{X})$	$\mathbf{X}_c, \mathbf{y}_c$
FITC (2005)	$\mathbf{U}$, $k(\mathbf{X}_c, \mathbf{U})$	$k(\mathbf{X}, \mathbf{U})$	$\subset \mathbf{X}_c, \mathbf{y}_c$
pFedGP + Global Predictor	$\mathbf{U}$, $k(\mathbf{X}_c, \mathbf{U})$	$k(\mathbf{X}, \mathbf{U})$	$\subset \mathbf{X}_c, \mathbf{y}_c$
Distributed GP (2015)	$\mathbf{x}_\star/\mathbf{\Phi}_\star$ or $\mathbf{X}_c/\mathbf{\Phi}_c$	$k(\mathbf{x}_\star, \mathbf{X}_c)$	$\mathbf{X}_c, \mathbf{y}_c$ or $\mathbf{X}_*, \mathbf{y}_*$
Distr. Variational GPs (2014; 2016)	$\mathbf{U}$	$k(\mathbf{U}, \cdot)$	$\subset \mathbf{X}_c$
SPAHM GP (2019)	$\mathbf{U}$	$k(\mathbf{U}, \cdot)$	$\subset \mathbf{X}_c$

Appendix A Privacy risks review: why DL or feature sharing is not private

We provide a more detailed discussion in this section to support our claim that directly using previous works about distributed GPs is not safe regarding privacy, and that global federated GP is challenging itself. In terms of privacy, the ultimate goal of FL is to prevent $\mathbf{X}_c$ and $\mathbf{y}_c$ from being (easily) recovered by any other entity, including the server, than their original owner c. We first analyze common risky messages by showing how to recover the raw data with them. We then discuss privacy leakage of different GP algorithms and summarize them in Table A.1. We will now discuss messages and algorithms listed in the table one by one.

A.1 Risky Messages

Features $\mathbf{X}_c$ ***or*** $\mathbf{\Phi}_c$

Without loss of generality, we can denote the prediction model by $y = f(\phi(\mathbf{x}))$, where $\phi(\mathbf{X}_c) = \mathbf{\Phi}_c$. Then the adversaries can simply compute $\mathbf{y}_c = f(\mathbf{\Phi}_c)$ to recover the targets of clients. This further extends to sharing $\mathbf{x}_\star$ or $\mathbf{\Phi}_\star$, as the testing points and their corresponding targets can also be confidential.

Inducing points $\mathbf{U}$ ***or*** $\mathbf{U}_c$

There are different ways of choosing inducing points. One way is to select a subset of the training data $\mathbf{X}$, and this clearly leaks raw data. Another way is to optimize them as hyperparameters. However, Smith et al. (2021) pointed out that inducing points optimized in this way often end up in regions of low data density and may correspond to outliers. In this case, there is a high risk that some inducing points will correspond directly to real data. They proposed to use differential privacy to protect the output,

but the input is left unprotected. Even in the case where inducing points lie in different spaces than the original input, it is still possible to partially recover real $\mathbf{y}_c$ with the same argument as $\mathbf{\Phi}_c$. Thus we want to refrain from using inducing points.

Cross kernel matrix $k(\mathbf{X}_c, \mathbf{U})$

In general, sharing $k(\mathbf{X}_c, \mathbf{U})$ does not reveal $\mathbf{X}_c$ immediately since the kernel function k tends to be non-invertible. However, in many algorithms that use inducing points, $k(\mathbf{X}_c, \mathbf{U})$ serves as sufficient statistics to predict $\mathbf{y}_c$. For example, for FITC (Snelson and Ghahramani, 2005), the predictive mean is given by:

$$
\begin{aligned}
f(\mathbf{x}_\star) &= k(\mathbf{x}_\star, \mathbf{U})\mathbf{\Sigma}k(\mathbf{U}, \mathbf{X})\mathbf{\Lambda}^{-1}\mathbf{y}, \text{ where} \\
\mathbf{\Sigma} &= (k(\mathbf{U}, \mathbf{U}) + k(\mathbf{U}, \mathbf{X})\mathbf{\Lambda}^{-1}k(\mathbf{X}, \mathbf{U}))^{-1} \\
\mathbf{\Lambda} &= diag[\mathbf{K} - k(\mathbf{X}, \mathbf{U})k(\mathbf{U}, \mathbf{U})^{-1}k(\mathbf{U}, \mathbf{X})]
\end{aligned}
$$

Then the adversaries can again retrieve $\mathbf{y}_c = k(\mathbf{X}_c, \mathbf{U})\mathbf{\Sigma}k(\mathbf{U}, \mathbf{X})\mathbf{\Lambda}^{-1}\mathbf{y}$ without knowing $\mathbf{X}_c$.

A.2 Algorithms

FedAvg

FedAvg shares no risky messages listed before. It shares model parameters only.

pFedGP

Similar to FedAvg, pFedGP only shares model hyperparameters. Any prediction that requires $\mathbf{K}, k(\mathbf{x}_\star, \mathbf{X})$ resorts to a local version $\mathbf{K}_c, k(\mathbf{x}_\star, \mathbf{X}_c)$.

FedBNR

In phase 1, we share hyperparameters like FedAvg and pFedGP, which is safe. In phase 2, we share hyperparameters and three other terms back and forth. No risky messages are shared.

- $\text{message}_1 := \mathbf{\Phi}_c\mathbf{\Phi}_c^\top$: The outer product (known as the scatter matrix) is a non-invertible function of $\mathbf{\Phi}_c$, and our model requires at least $\mathbf{\Phi}_c$ to predict $\mathbf{y}_c$.
- $\text{message}_2 := \mathbf{\Phi}_c\mathbf{y}_c$: This alone poses an under-constrained system of equations that has m equations and $(m+1)n_c$ variables, so $\mathbf{\Phi}_c, \mathbf{y}_c$ is non-identifiable.
- $\text{message}_3 := \bar{\mathbf{w}} = \sigma^{-2}(\sigma^{-2}\sum_{c\in S}\mathbf{\Phi}_c\mathbf{\Phi}_c^\top + \lambda^{-2}\mathbf{I})^{-1}\sum_{c\in S}(\mathbf{\Phi}_c\mathbf{y}_c)$: This alone poses an under-constrained system of equations that has m equations and $(m+1)n$ variables, so $\mathbf{\Phi}, \mathbf{y}$ is non-identifiable.

In total, these three messages pose a system of equations that has $m(m+1)/2 + 2m = 0.5m^2 + 2.5m$ equations and $(m+1)n$ variables (i.e. $\mathbf{\Phi}, \mathbf{y}$). Since $m \ll n$, meaning the number of random samples is much smaller than the size of training data, the system has multiple solutions and the true values of $\mathbf{\Phi}, \mathbf{y}$ are non-identifiable. Similarly, the first two messages pose a (local) system of equations that has $m(m+1)/2 + m = 0.5m^2 + 1.5m$ equations and $(m+1)n_c$ variables (i.e. $\mathbf{\Phi}_c, \mathbf{y}_c$). As long as $1 < m \leq n_c$, the variables are still non-identifiable.

Centralized GP

The predictive mean is given by:

$$f(\mathbf{x}_\star) = k(\mathbf{x}_\star, \mathbf{X})(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}$$

Thus, even if we can learn the hyperparameters in a FL fashion without privacy leakage, we have to share $\mathbf{X}_c/\mathbf{\Phi}_c$ to calculate $\mathbf{K}$ and $k(\mathbf{x}_\star, \mathbf{X})$.

FITC

FITC has to share $k(\mathbf{X}_c, \mathbf{U})$ to calculate the term $k(\mathbf{X}, \mathbf{U})$ in $\mathbf{\Sigma}$. See FITC in the paragraph of Cross kernel matrix $k(\mathbf{X}_c, \mathbf{U})$ for details. Also, inducing points $\mathbf{U}$ have to be shared with all clients.

pFedGP + Global Predictor

Following the arguments with centralized GP and inducing point approximation, pFedGP cannot generalize to a global predictor without sharing $\mathbf{X}_c/\mathbf{\Phi}_c$ (without approximation) or $\mathbf{U}$ and $k(\mathbf{X}_c, \mathbf{U})$ (with the FITC approximation they adapt).

Distributed GP

Deisenroth and Ng (2015) introduces a collection of works that uses the Product of Experts (PoE) model to approximate the global GP with local GPs. In the most basic setting, the global predictive mean and variance are given by:

$$\mu(\mathbf{x}_\star) = \sigma^2(\mathbf{x}_\star)\sum_{c\in S}\sigma_c^{-2}(\mathbf{x}_\star)\mu_c(\mathbf{x}_\star)$$
$$\sigma^{-2}(\mathbf{x}_\star) = \sum_{c\in S}\sigma_c^{-2}(\mathbf{x}_\star)$$

where $\mu_c(\mathbf{x}_\star)$ and $\sigma_c^2(\mathbf{x}_\star)$ are the local predictive mean and variance. Then, to calculate these two terms, we either have to share $\mathbf{x}_\star/\mathbf{\Phi}_\star$ or collect all the client experts for the tester to perform global prediction locally, where a client expert is a stand-alone GP predictive model with local training data. However, collecting client experts requires sharing $\mathbf{X}_c/\mathbf{\Phi}_c$, by the formulas of predictive mean and variance of GP. Note all the other advanced methods proposed by Deisenroth and Ng (2015) have similar structures and thus are not safe.

Distributed Variational GPs

These works are also based on inducing points, which may leak real data when clients have low data density regions, which is common for high dimensional data.

SPAHM GP

This paper did not specify technical details about the GP predictor they learned, so it remains unclear if there are other risky messages sent during the process, but they mentioned they used inducing points to learn GPs, which needs to be avoided in FL.

A.3 A differential privacy (DP) argument

In this section, we show why sharing kernels is less private than sharing scatter matrices by examining how much noise is needed with DP.

The amount of noise needed with DP significantly increases when the information released has size that is dependent of the size of data, when the released information contains non-constant interaction terms between records. This often trashes the utility of the trained model, suggesting that such messages themselves contain a lot information about the raw data and are prone to privacy bleaching. As a concrete comparison, we can calculate the sensitivity of $f_k(\mathbf{X}) = \Phi(\mathbf{X})^\top \Phi(\mathbf{X})$, the $n \times n$ sized kernel, and $f_s(\mathbf{X}) = \Phi(\mathbf{X})\Phi(\mathbf{X})^\top$, the $m \times m$ sized scatter matrix. For simplicity, use $\Phi(\mathbf{X}) = \sin(\mathbf{X}), \forall \mathbf{X} \in \mathbb{R}^{n \times m}$ so that its sensitivity is bounded. W.l.o.g, let $\mathbf{X}' = \mathbf{X}[0 : n-1, :]$ be the first $n-1$ records of $\mathbf{X}$. We pad $f_k(\mathbf{X}')$ with 0's so that $f_k(\mathbf{X})$ and $f_k(\mathbf{X}')$ are then both of size $n \times n$. Then,

$$
\begin{aligned}
f_k(\mathbf{X}) - f_k(\mathbf{X}') &= \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_{n-1}) & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & \cdots & k(\mathbf{x}_2, \mathbf{x}_{n-1}) & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \ddots & \vdots & \vdots \\ k(\mathbf{x}_{n-1}, \mathbf{x}_1) & \cdots & k(\mathbf{x}_{n-1}, \mathbf{x}_{n-1}) & k(\mathbf{x}_{n-1}, \mathbf{x}_n) \\ k(\mathbf{x}_n, \mathbf{x}_1) & \cdots & k(\mathbf{x}_n, \mathbf{x}_{n-1}) & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix} \\
&\quad - \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_{n-1}) & 0 \\ k(\mathbf{x}_2, \mathbf{x}_1) & \cdots & k(\mathbf{x}_2, \mathbf{x}_{n-1}) & 0 \\ \vdots & \ddots & \vdots & \vdots \\ k(\mathbf{x}_{n-1}, \mathbf{x}_1) & \cdots & k(\mathbf{x}_{n-1}, \mathbf{x}_{n-1}) & 0 \\ 0 & \cdots & 0 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & \cdots & 0 & k(\mathbf{x}_1, \mathbf{x}_n) \\ 0 & \cdots & 0 & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & k(\mathbf{x}_{n-1}, \mathbf{x}_n) \\ k(\mathbf{x}_n, \mathbf{x}_1) & \cdots & k(\mathbf{x}_n, \mathbf{x}_{n-1}) & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}
\end{aligned}
$$

where $k(\mathbf{x}_i, \mathbf{x}_j) = \sin(\mathbf{x}_i)^\top \sin(\mathbf{x}_j)$. We flatten the above matrix to a vector and calculate the L_2 sensitivity. There are $2n-1$ non-zero terms in the above result. Note $k(\mathbf{x}_i, \mathbf{x}_j) \leq m, \forall i, j$. By definition,

$$
\begin{aligned}
L_2(f_k) &= \max_{\mathbf{X}, \mathbf{X}'} ||\text{flatten}(f_k(\mathbf{X}) - f_k(\mathbf{X}'))||_2 \\
&= |||[\underbrace{0, \cdots, 0}_{(n-1)^2 \text{ terms}}, \underbrace{m, \cdots, m}_{2n-1 \text{ terms}}]||_2 \\
&= \sqrt{2n-1}m
\end{aligned}
$$

On the other hand, for scatter matrices,

$$\begin{aligned} f_s(\mathbf{X}) - f_s(\mathbf{X}') &= \sum_{i=1}^{n} \sin(\mathbf{x}_i)\sin(\mathbf{x}_i)^\top - \sum_{i=1}^{n-1} \sin(\mathbf{x}_i)\sin(\mathbf{x}_i)^\top \\ &= \sin(\mathbf{x}_n)\sin(\mathbf{x}_n)^\top \\ L_2(f_s) &= \max_{\mathbf{X},\mathbf{X}'} ||\text{flatten}(f_s(\mathbf{X}) - f_s(\mathbf{X}'))||_2 \\ &= \max_{\mathbf{x}} ||\text{flatten}(\sin(\mathbf{x})\sin(\mathbf{x})^\top)||_2 \\ &= |||\underbrace{[1, \cdots, 1]}_{m^2 \text{ terms}}||_2 \\ &= m \end{aligned}$$

Since $L_2(f_k)$ grows with the size of data, the magnitude of noise needed to make f_k differentially private also grows to infinity, while $L_2(f_s)$ stays a small constant $m \ll \sqrt{2n-1}m$.

A.4 Variance adversarial attack

We acknowledge that uncertainty quantification could pose potential negative consequences for privacy. For example, it is well-known that with stationary kernels, the variance of the GP predictive posterior tends to form local minima at places where training points locate, since by definition the variance of stationary kernels depends on the distance to the training points. In this case, it is possible to at least retrieve the target $\mathbf{y}$ by searching for the local minima of the variance. However, this property does not hold true in general when we work with non-stationary deep kernels, where the variance depends on the input locations. Figure A.1 shows an example of FedBNR where the variance adversarial attack fails. The left graph shows the "real" training data we wish to protect, and the corresponding predictive posterior learned by FedBNR with a deep kernel. The right graph shows the amount of variance for the prediction in the left graph. We can see the local minima of predictive variance are reasonably far away from training data, and thus the variance adversarial attack described above fails to retrieve any training data.

A.5 FedBNR+DP: sensitivities

To use the Gaussian mechanism to protect FedBNR, we need to add noise to all the messages sent by the clients. In the first phase, kernel hyperparameters are revealed. In the second phase, $\mathbf{\Phi}_c\mathbf{\Phi}_c^\top$, $\mathbf{\Phi}_c\mathbf{y}_c$, and $\mathbf{y}_c^\top\mathbf{y}_c$ are revealed. The L_2 sensitivity of each function is as follows:

- In phase 1, L_2 sensitivity of the hyperpameters is the L_2 gradient clipping constant times the number of local update rounds times the learning rate ζ. Since we take the average during aggregation, the actual sensitivity of the mean has an extra factor of $1/\sqrt{n_S}$, where n_S is the number of clients. This means the more clients there are in the system, the less perturbation the mean exhibits. This factor further

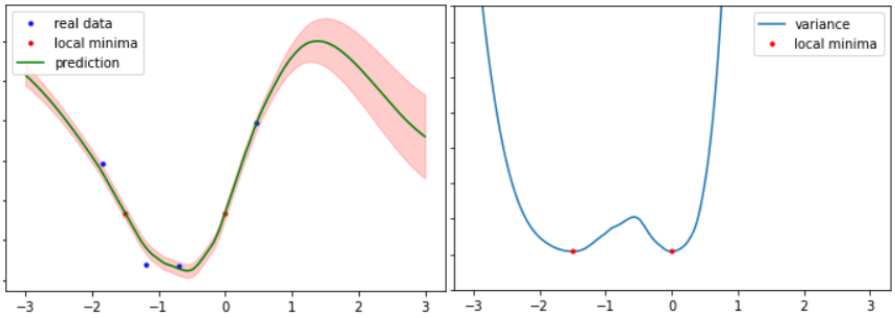

Fig. A.1 An example to show why variance adversarial attack fails when we have a non-stationary kernel. Left: training data and the predictive posterior of FedBNR. Right: the amount of variance for the prediction in the left graph.

decreases to $1/n_S$ if we adapt secure multi-party computation and add noise directly to the summation of all the messages globally, instead of adding noise to messages at clients.

- In phase 2, suppose we use the latent stationary architecture ($\sin, \cos$ activation functions normalized by $\sqrt{m}$), then each entry of $\mathbf{\Phi}_c$ can change by up to $2/\sqrt{m}$. $\mathbf{y}$ is in general unbounded, so we need to clip it by its max and min, and denote $b := max(\mathbf{y}) - min(\mathbf{y})$. Then we can flatten $s(\mathbf{X}_c, \mathbf{y}_c) = [\mathbf{\Phi}_c\mathbf{\Phi}_c^\top, \mathbf{\Phi}_c\mathbf{y}_c, \mathbf{y}_c^\top\mathbf{y}_c]$ and calculate its sensitivity:

$$L_2(s) = \sqrt{m(2m+1)(4/m)^2 + 2m(2b/\sqrt{m})^2 + b^4} \tag{A1}$$

$$= \sqrt{32 + 16/m + 8b^2 + b^4} \tag{A2}$$

Note $\mathbf{\Phi}_c\mathbf{\Phi}_c^\top$ is symmetric and we only need to protect the upper half triangular matrix, and this architecture results in $2m$ random features with m samples.

Appendix B Experiment details

Figure A.2 shows the architecture of DNNs used in the real world regression experiments. For FedAvg, FedProx, and pFedGP, we used the left DNN as their feature extractor. For our methods, we used the rightmost architecture in Figure 2 with the same feature extractor f, a very small distribution shifter h (the right DNN in Figure A.2), and $\omega \sim N(\mathbf{0}, \mathbf{I}_5)$.

We run each random seed of each dataset on 1 CPU and 1 NVIDIA T4 GPU with 16GB RAM. Some important hyperparameters are listed in Table A.2. These hyperparameters are selected through grid searching, as suggested by FedProx. We make our code public in the supplementary materials, where other minutiae like the learning rate can be found. All datasets included along with our code are licensed under a Creative Commons Attribution 4.0 International (CC BY 4.0) license, which allows for the sharing and adaptation of the datasets for any purpose, provided that the appropriate credit is given.

Table A.2 Hyperparameters used in the real world regression experiments. For FedLoc $L = 10\rho$ to run the Proximal ADMM algorithm.

	Skillcraft		SML		Parkinsons		Bike		CCPP	
#clients	10	100	10	100	10	100	10	100	10	100
size s	200		2000		2000		5000		5000	
FedProx μ	0.5	1.0	0.1	1.0	1.0	1.0	0.1	0.01	0.001	1.0
FedLoc ρ	5e3	5e4	5e3	5e4	1e3	5e3	5e4	5e4	1e5	1e5
FedBNR-KD α	10	2	1	0.5	5	2	5	0.5	5	5

Appendix C Additional results

C.1 Necessity of random features

We perform an extra ablation study experiment similarly to the real-world experiment in generalization setting. Experiment details are the same as in Appendix B. Each setting is run for 10 different random seeds. RMSE reported in Table A.3 shows that random features significantly improve the model capacity.

C.2 Generalization setting

We assess models with testing data uniformly sampled globally in this setting.

We include the maximum calibration error (MCE) and the Brier score (BRI) of the real world regression experiments in Table A.4 and A.5. MCE measures the (estimated) worst difference between $p\%$ confidence intervals and $p'\%$ test points falling into these intervals. BRI measures the mean squared difference between the confidence and observations, where a test point falling in the CI counts as 1, otherwise 0. We also include the standard error of the mean (SEM) in Table A.6. Our methods still perform better in most of the cases.

C.3 Differential Privacy

We report the trade-off between model utility and the privacy budget ϵ in Figure A.3. The y-axis reports RMSE, and the light area reports the standard error of the mean (SEM). Experiments were repeated for 5 different seeds. Other experiment details are similar to the generalization setting.

C.4 Personalization setting

We assess models with testing data uniformly sampled locally in this setting.

We report MCE in Table A.7 and BRI in Table A.8.

Appendix D Unifying random kernel

D.1 Approximate deep stationary kernels

We now prove Thm. 2 in the main paper. We make use of the Hoeffding's inequality (Hoeffding, 1994).

Theorem A.1 (Hoeffding's inequality)**.** *Suppose $X_1, \cdots, X_n$ are n independent random variables. Let $\bar{X} = \sum_{i=1}^n X_i/n$. If $\forall i \in \{1, \cdots, n\}, \Pr(X_i \in [a_i, b_i]) = 1$ almost surely, then $\forall t > 0$*

$$\Pr[|\bar{X} - \mathbb{E}[\bar{X}]| \geq t] \leq 2\exp\left(-\frac{2t^2n^2}{\sum_{i=1}^n (b_i - a_i)^2}\right)$$

Theorem 2. *$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^p, \forall p(\omega)$ being a proper distribution over $\mathbb{R}^{d'}$, and $g : \mathbb{R}^{d'} \times \mathbb{R}^p \to \mathbb{R}^d$ has the form of Eq. 15. With $m \in \mathbb{Z}_+$ samples of ω, $\forall t > 0$*

$$\Pr[|URK^m_{\omega,g}(\mathbf{x}, \mathbf{x}') - k(\mathbf{x}, \mathbf{x}')| \geq t] \leq 2\exp\left(-\frac{mt^2}{8}\right)$$

Proof. Notice $\mathrm{URK}^m_{\omega,g}(\mathbf{x}, \mathbf{x}')$ is the empirical estimation of $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}_\omega[g(\omega, \mathbf{x})^\top g(\omega, \mathbf{x}')]$:

$$\mathrm{URK}^m_{\omega,g}(\mathbf{x}, \mathbf{x}') = \left(\frac{\mathbf{s}^m_\omega(\mathbf{x})}{\sqrt{m}}\right)^\top \left(\frac{\mathbf{s}^m_\omega(\mathbf{x}')}{\sqrt{m}}\right) = \sum_{i=1}^m \frac{1}{m}(g(\omega_i, \mathbf{x})^\top g(\omega_i, \mathbf{x}'))$$

$g(\omega_i, \mathbf{x})^\top g(\omega_i, \mathbf{x}')$ are i.i.d. 1-dimensional random variables. It is bounded by $[-2, 2]$ since $g(\boldsymbol{\omega}, \mathbf{x}) = [\sin(\boldsymbol{\omega}^\top \phi_{\mathrm{NN}}(\mathbf{x})), \cos(\boldsymbol{\omega}^\top \phi_{\mathrm{NN}}(\mathbf{x}))]^\top$. Finally, apply the Hoeffding's inequality. □

D.2 Approximate non-stationary kernels without DNN

Additionally, URK can be used to construct feature mappings of non-stationary kernels without the help of Deep Neural Networks. We prove the two examples we gave in the main paper now.

Proposition 3. *Let $\omega \sim N(\mathbf{0}, \mathbf{I}), g(\omega, \mathbf{x}) = \exp(\omega^\top \mathbf{x})$, then $k(\mathbf{x}, \mathbf{x}') = \lim_{m\to\infty} URK_{\omega,g}(\mathbf{x}, \mathbf{x}')$ has infinite features.*

Proof.

$$\begin{aligned}
k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}_\omega[\mathbf{z}_\omega(\mathbf{x})^\top \mathbf{z}_\omega(\mathbf{x}')] \\
&= \mathbb{E}_\omega[\exp\left(\omega^\top(\mathbf{x} + \mathbf{x}')\right)] \\
&= M_\omega(\mathbf{x} + \mathbf{x}'), \text{ the moment generating function of } \omega \\
&= \exp((\mathbf{x} + \mathbf{x}')^\top(\mathbf{x} + \mathbf{x}')/2) \\
&= \exp((\mathbf{x}^\top\mathbf{x} + \mathbf{x}'^\top\mathbf{x}')/2)\exp(\mathbf{x}^\top\mathbf{x}')
\end{aligned}$$

$$
\begin{aligned}
&= \exp((\mathbf{x}^\top \mathbf{x} + \mathbf{x}'^\top \mathbf{x}')/2) \sum_{l=0}^{\infty} \frac{(\mathbf{x}^\top \mathbf{x}')^l}{l!} \\
&= \sum_{l=0}^{\infty} \left(\frac{\exp(\mathbf{x}^\top \mathbf{x}/2)}{\sqrt{l!}} (\mathbf{x}^\top \mathbf{x}')^l \frac{\exp(\mathbf{x}'^\top \mathbf{x}'/2)}{\sqrt{l!}} \right) \\
&= \phi(\mathbf{x})^\top \phi(\mathbf{x}')
\end{aligned}
$$

□

Theorem 4. *Let ω and g have the form of Prop. 3.2. Suppose $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^p$ satisfies $||\mathbf{x}||_2^2 \leq a, ||\mathbf{x}'||_2^2 \leq a'$. With $m \in \mathbb{Z}_+$ samples of ω, $\forall t \in (0, \exp(a + a')/2))$, by Fenton-Wilkinson approximation,*

$$
\begin{aligned}
\sigma_s := \log \left(\frac{\exp(a + a') - 1}{m} + 1 \right), E := \exp((a + a')/2) \\
\Pr[|URK_{\omega,g}^m(\mathbf{x}, \mathbf{x}') - k(\mathbf{x}, \mathbf{x}')| \geq t] \leq 1 - \boldsymbol{F}\left(\frac{2\log(t + E) - (a + a') + \sigma_s^2}{2\sigma_s} \right) \\
+ \boldsymbol{F}\left(\frac{2\log(E - t) - (a + a') + \sigma_s^2}{2\sigma_s} \right)
\end{aligned}
$$

$\boldsymbol{F}$ is the cumulative probability distribution function of the standard normal distribution.

Proof. Since $\omega_i \sim N(\mathbf{0}, \mathbf{I}), \forall i$ and $g(\omega_i, \mathbf{x})^\top g(\omega_i, \mathbf{x}') = \exp(\omega_i^\top (\mathbf{x} + \mathbf{x}'))$,

$$
\omega_i^\top (\mathbf{x} + \mathbf{x}') \sim N(0, \sigma^2), \sigma^2 \leq a + a'
$$

$$
g(\omega_i, \mathbf{x})^\top g(\omega_i, \mathbf{x}') \sim LogN(0, \sigma^2), \text{ the log-normal distribution}
$$

Since the sum of m i.i.d. log-normal distributions has no closed form, we use the Fenton-Wilkinson approximation (Fenton, 1960) to model $\mathrm{URK}_{\omega,g}^m(\mathbf{x}, \mathbf{x}')$. It approximates the sum with a log-normal distribution by matching the first and second moments.

$$
\begin{aligned}
\sigma_s^2 = \log \left(\frac{\exp(\sigma^2) - 1}{m} + 1 \right), \; E := k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[g(\omega, \mathbf{x})^\top g(\omega, \mathbf{x}')] = \exp(\sigma^2/2) \\
\mathrm{URK}_{\omega,g}^m(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{m} \frac{1}{m} (g(\omega_i, \mathbf{x})^\top g(\omega_i, \mathbf{x}')) \sim LogN((\sigma^2 - \sigma_s^2)/2, \sigma_s^2) \\
\Pr[\mathrm{URK}_{\omega,g}^m(\mathbf{x}, \mathbf{x}') - E \geq t] = 1 - \Pr[\mathrm{URK}_{\omega,g}^m(\mathbf{x}, \mathbf{x}') \leq t + E] \\
= 1 - \boldsymbol{F}\left(\frac{2\log(t + E) - \sigma^2 + \sigma_s^2}{2\sigma_s} \right) \\
\Pr[\mathrm{URK}_{\omega,g}^m(\mathbf{x}, \mathbf{x}') - E \leq -t] = \Pr[\mathrm{URK}_{\omega,g}^m(\mathbf{x}, \mathbf{x}') \leq E - t] \\
= \boldsymbol{F}\left(\frac{2\log(E - t) - \sigma^2 + \sigma_s^2}{2\sigma_s} \right)
\end{aligned}
$$

$$\Pr[|\mathrm{URK}^m_{\omega,g}(\mathbf{x},\mathbf{x}') - k(\mathbf{x},\mathbf{x}')| \geq t] = \Pr[\mathrm{URK}^m_{\omega,g}(\mathbf{x},\mathbf{x}') - E \geq t] + \Pr[\mathrm{URK}^m_{\omega,g}(\mathbf{x},\mathbf{x}') - E \leq -t]$$

This equation takes its maximum when $\sigma^2 = a + a'$. □

Proposition 5. *Let* $\tilde{c}, \tilde{n} \in \mathbb{R}$. *Define* $\tilde{\mathbf{p}} = [\frac{1}{2}, \frac{1}{2p}, \frac{1}{2p}, \cdots, \frac{1}{2p}]^\top \in \mathbb{R}^{p+1}$, $\omega \sim Multi(\tilde{n}, \tilde{\mathbf{p}})$, *the multinomial distribution. Define* $\tilde{\mathbf{x}} = [\sqrt{2\tilde{c}}, \sqrt{2p}\mathbf{x}^\top]^\top \in \mathbb{R}^{p+1}$, $g(\omega, \mathbf{x}) = \exp(\omega^\top \log \tilde{\mathbf{x}})$, *then* $k(\mathbf{x},\mathbf{x}') = \lim_{m\to\infty} URK_{\omega,g}(\mathbf{x},\mathbf{x}') = (\mathbf{x}^\top\mathbf{x}' + \tilde{c})^{\tilde{n}}$

Proof. In the following proof, any subscript i means the i_{th} entry of the vector. By definition,

$$\begin{aligned}
k(\mathbf{x},\mathbf{x}') &= \mathbb{E}_\omega[\mathbf{z}_\omega(\mathbf{x})^\top \mathbf{z}_\omega(\mathbf{x}')] \\
&= \mathbb{E}_\omega[\exp\left(\omega^\top(\log\tilde{\mathbf{x}} + \log\tilde{\mathbf{x}}')\right)] \\
&= M_\omega(\log\tilde{\mathbf{x}} + \log\tilde{\mathbf{x}}'), \text{ the moment generating function of } \omega \\
&= \left(\sum_{i=1}^{p+1} \mathbf{p}_{poly,i} \exp(\log\tilde{\mathbf{x}}_i + \log\tilde{\mathbf{x}}'_i)\right)^{\tilde{n}} \\
&= \left(\frac{1}{2}\exp(2\log\sqrt{2\tilde{c}}) + \sum_{i=1}^{p} \frac{1}{2p}\exp(\log\sqrt{2p}\mathbf{x}_i + \log\sqrt{2p}\mathbf{x}'_i)\right)^{\tilde{n}} \\
&= \left(\tilde{c} + \sum_{i=1}^{p} \mathbf{x}_i\mathbf{x}'_i\right)^{\tilde{n}} \\
&= (\mathbf{x}^\top\mathbf{x}' + \tilde{c})^{\tilde{n}}
\end{aligned}$$

□

Theorem 6. *Let* ω *and* g *have the form of Prop. 3.4. Suppose entries of* $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^p_+$ *are bounded by* $[a, b]$. *With* $m \in \mathbb{Z}_+$ *samples of* ω, $\forall t > 0$,

$$v := \frac{max\{\sqrt{2\tilde{c}}, \sqrt{2p}b\}}{min\{\sqrt{2\tilde{c}}, \sqrt{2p}a\}}$$

$$\Pr[|\, URK^m_{\omega,g}(\mathbf{x},\mathbf{x}') - k(\mathbf{x},\mathbf{x}')| \geq t] \leq 2\exp\left(-\frac{2mt^2}{v^{4\tilde{n}}}\right)$$

Proof. Similarly to Thm. 2, $g(\omega_i, \mathbf{x})^\top g(\omega_i, \mathbf{x}')$ are i.i.d. 1-dimensional random variables, bounded by $[\exp(\tilde{n}\log(\min\{\sqrt{2\tilde{c}}, \sqrt{2p}a\})), \exp(\tilde{n}\log(\max\{\sqrt{2\tilde{c}}, \sqrt{2p}b\}))]$. Then, apply Hoeffding's inequality. □

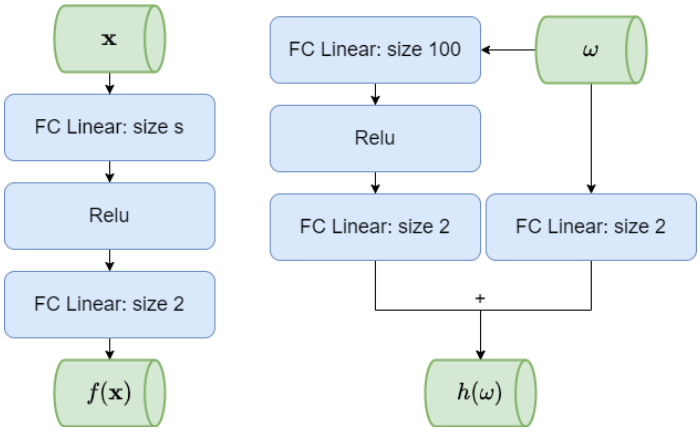

Fig. A.2 Architecture of DNNs used in real world regression experiments. Left: the feature extractor for $\mathbf{x}$; right: the distribution shifter for ω. FC is short for "fully connected". The layer size s varies for different datasets.

Table A.3 RMSE reported. Generalization setting. FedBNR\URK: removing random features from FedBNR; FedBNR: ordinary FedBNR. $\Uparrow$ and $\uparrow$ denote significantly worse results with $p < 0.01$ and $p < 0.05$ respectively; $\Downarrow$ and $\downarrow$ denote significantly better results similarly.

	Skillcraft		SML		Parkinsons		Bike		CCPP	
#clients	10	100	10	100	10	100	10	100	10	100
FedBNR\URK	$1.15^{\Uparrow}$	$1.20^{\Uparrow}$	$0.26^{\uparrow}$	$0.64^{\Uparrow}$	$5.95^{\Uparrow}$	$6.72^{\Uparrow}$	**0.40**	$0.43^{\Uparrow}$	**4.23**$^{\Downarrow}$	**4.16**$^{\Downarrow}$
FedBNR	**1.00**	**0.97**	**0.24**	**0.52**	**3.16**	**5.33**	**0.40**	**0.41**	4.41	4.51

Table A.4 Maximum calibration error (MCE) reported. Generalization setting. $\Uparrow$ and $\uparrow$ denote significantly worse results with $p < 0.01$ and $p < 0.05$ respectively; $\Downarrow$ and $\downarrow$ denote significantly better results similarly.

	Skillcraft		SML		Parkinsons		Bike		CCPP	
#clients	10	100	10	100	10	100	10	100	10	100
Central GP	0.03		0.16		0.42		0.16		0.40	
pFedGP	$0.77^{\Uparrow}$	$0.65^{\Uparrow}$	$0.82^{\Uparrow}$	$0.83^{\Uparrow}$	$0.93^{\Uparrow}$	$0.91^{\Uparrow}$	$0.78^{\Uparrow}$	$0.73^{\Uparrow}$	$0.94^{\Uparrow}$	$0.93^{\Uparrow}$
FedLoc	$0.51^{\Uparrow}$	$0.81^{\Uparrow}$	**0.20**$^{\Downarrow}$	$0.44^{\Uparrow}$	$0.54^{\Uparrow}$	$0.76^{\Uparrow}$	$0.42^{\Uparrow}$	$0.28^{\Uparrow}$	$0.39^{\Downarrow}$	$0.95^{\Uparrow}$
DEC-apx	$0.95^{\Uparrow}$	$0.80^{\Uparrow}$	$0.95^{\Uparrow}$	$0.95^{\Uparrow}$	$0.95^{\Uparrow}$	$0.95^{\Uparrow}$	$0.34^{\Uparrow}$	$0.29^{\Uparrow}$	$0.74^{\Uparrow}$	$0.65^{\Uparrow}$
DEC-gapx	$0.95^{\Uparrow}$	$0.73^{\Uparrow}$	$0.95^{\Uparrow}$	$0.95^{\Uparrow}$	$0.95^{\Uparrow}$	$0.95^{\Uparrow}$	$0.44^{\Uparrow}$	$0.32^{\Uparrow}$	$0.84^{\Uparrow}$	$0.75^{\Uparrow}$
ours										
FedBNR	**0.09**	$0.32^{\Uparrow}$	$0.67^{\Uparrow}$	$0.62^{\Uparrow}$	$0.62^{\Uparrow}$	$0.72^{\Uparrow}$	**0.15**	$0.64^{\Uparrow}$	**0.39**$^{\Downarrow}$	**0.31**$^{\Downarrow}$
FedBNR-KD	0.10	**0.11**	0.33	**0.35**	**0.47**	**0.48**	**0.15**	**0.17**	0.49	0.50

Table A.5 Brier score (BRI) reported. Generalization setting. $\Uparrow$ and $\uparrow$ denote significantly worse results with $p < 0.01$ and $p < 0.05$ respectively; $\Downarrow$ and $\downarrow$ denote significantly better results similarly.

	Skillcraft		SML		Parkinsons		Bike		CCPP	
#clients	10	100	10	100	10	100	10	100	10	100
Central GP	0.16		0.20		0.24		0.18		0.24	
pFedGP	$0.30^{\Uparrow}$	$0.28^{\Uparrow}$	$0.31^{\Uparrow}$	$0.31^{\Uparrow}$	$0.33^{\Uparrow}$	$0.33^{\Uparrow}$	$0.30^{\Uparrow}$	$0.30^{\Uparrow}$	$0.33^{\Uparrow}$	$0.33^{\Uparrow}$
FedLoc	0.21	$0.31^{\Uparrow}$	**0.18**$^{\Downarrow}$	$0.25^{\Uparrow}$	$0.26^{\Uparrow}$	$0.30^{\Uparrow}$	$0.19^{\downarrow}$	$0.16^{\Downarrow}$	$0.24^{\Downarrow}$	$0.33^{\Uparrow}$
DEC-apx	$0.33^{\Uparrow}$	$0.31^{\Uparrow}$	$0.33^{\Uparrow}$	$0.33^{\Uparrow}$	$0.33^{\Uparrow}$	$0.33^{\Uparrow}$	**0.17**$^{\Downarrow}$	**0.15**$^{\Downarrow}$	$0.27^{\Uparrow}$	$0.24^{\Downarrow}$
DEC-gapx	$0.33^{\Uparrow}$	$0.28^{\Uparrow}$	$0.33^{\Uparrow}$	$0.33^{\Uparrow}$	$0.33^{\Uparrow}$	$0.33^{\Uparrow}$	$0.20^{\Uparrow}$	$0.26^{\Uparrow}$	$0.30^{\Uparrow}$	0.26
ours										
FedBNR	**0.18**	$0.22^{\Uparrow}$	$0.29^{\Uparrow}$	$0.28^{\Uparrow}$	$0.28^{\Uparrow}$	$0.30^{\Uparrow}$	$0.20^{\Uparrow}$	$0.29^{\Uparrow}$	**0.24**$^{\Downarrow}$	**0.22**$^{\Downarrow}$
FedBNR-KD	**0.18**	**0.18**	0.23	**0.23**	**0.25**	**0.26**	0.19	0.20	0.25	0.26

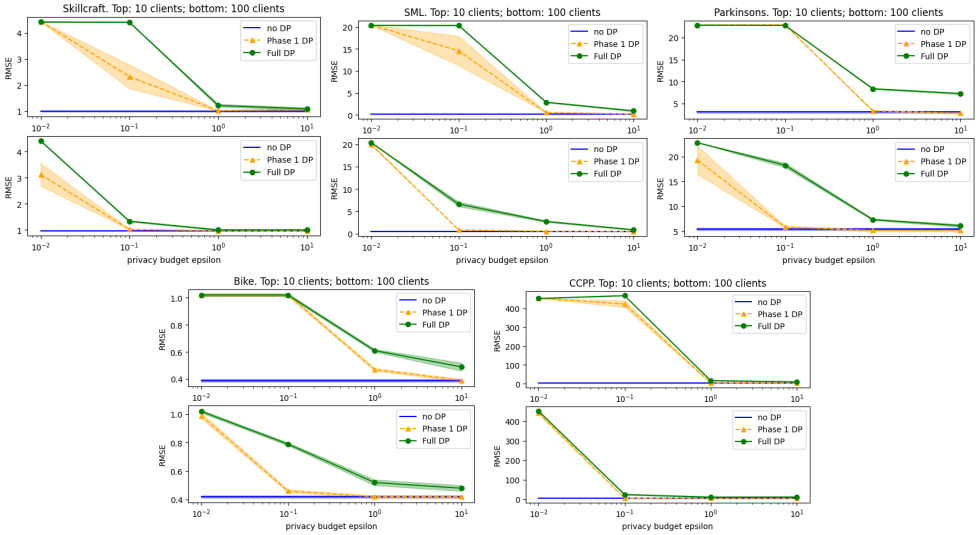

Fig. A.3 FedBNR+DP results. Phase 1 DP: add DP protection to model parameters θ only; full DP: add DP protection to all messages.

	Skillcraft		SML		Parkinsons		Bike		CCPP	
#clients	10	100	10	100	10	100	10	100	10	100
local+local	1.26±0.03	1.48±0.01	1.49±0.06	2.32±0.03	10.9±0.26	10.6±0.05	0.79±0.02	0.91±0.01	14.6±0.31	19.3±2.08
local+global	1.08±0.02	1.22±0.01	1.00±0.02	1.65±0.01	6.42±0.10	7.42±0.01	0.59±0.01	0.73±0.01	5.62±0.19	7.03±0.16
avg+local	1.05±0.02	1.06±0.02	0.61±0.06	0.81±0.05	9.84±0.31	8.80±0.11	0.45±0.01	0.54±0.01	8.32±0.43	13.4±0.43
kd+local	1.04±0.01	1.28±0.02	0.86±0.06	1.34±0.12	6.15±0.30	6.97±0.27	0.51±0.01	0.61±0.01	10.0±0.61	17.1±0.63
FedAvg	1.00±0.01	1.03±0.01	0.36±0.02	0.71±0.02	6.83±0.16	7.38±0.21	**0.38±0.01**	0.42±0.01	4.45±0.06	4.44±0.04
FedProx	0.98±0.01	1.05±0.01	0.34±0.01	0.62±0.02	6.32±0.27	7.38±0.25	0.39±0.01	0.42±0.01	4.43±0.04	4.47±0.02
pFedGP	0.99±0.01	1.15±0.01	0.75±0.05	1.34±0.04	9.36±0.16	8.91±0.09	0.45±0.02	0.46±0.01	14.2±0.61	18.2±0.23
FedLoc	1.08±0.01	4.15±0.26	2.83±0.07	5.43±0.04	8.40±0.02	11.5±0.03	0.64±0.01	0.75±0.01	20.6±0.74	44.1±0.61
DEC-apx	1.10±0.01	4.34±0.01	1.90±0.04	4.93±0.08	8.59±0.07	9.21±0.03	0.76±0.01	0.88±0.01	7.64±0.28	11.06±0.14
DEC-gapx	**0.94±0.01**	4.31±0.01	1.41±0.01	3.57±0.09	8.07±0.02	8.26±0.01	0.55±0.01	0.75±0.01	6.26±0.04	9.12±0.22
ours										
FedBNR	0.98±0.01	**0.97±0.01**	**0.25±0.01**	**0.44±0.02**	**3.10±0.23**	5.42±0.20	0.39±0.01	**0.42±0.01**	4.40±0.01	4.51±0.03
FedBNR-KD	0.96±0.01	0.98±0.01	0.55±0.01	0.55±0.01	4.58±0.05	**4.67±0.04**	0.43±0.01	0.48±0.01	**4.38±0.01**	**4.38±0.01**

Table A.6 RMSE ± standard error of the mean (SEM) reported. Generalization setting.

Table A.7 Maximum calibration error (MCE) reported. Personalization setting. $\Uparrow$ and $\uparrow$ denote significantly worse results with $p < 0.01$ and $p < 0.05$ respectively; $\Downarrow$ and $\downarrow$ denote significantly better results similarly.

	Skillcraft		SML		Parkinsons		Bike		CCPP	
#clients	10	100	10	100	10	100	10	100	10	100
pFedGP	$0.76^{\Uparrow}$	$0.59^{\Uparrow}$	$0.82^{\Uparrow}$	$0.80^{\Uparrow}$	$0.83^{\Uparrow}$	$0.81^{\Uparrow}$	$0.77^{\Uparrow}$	$0.68^{\Uparrow}$	$0.93^{\Uparrow}$	$0.93^{\Uparrow}$
pxADMM	$0.76^{\Uparrow}$	$0.58^{\Uparrow}$	$0.94^{\Uparrow}$	$0.83^{\Uparrow}$	$0.93^{\Uparrow}$	$0.81^{\Uparrow}$	$0.30^{\Uparrow}$	$0.34^{\Uparrow}$	$0.94^{\Uparrow}$	$0.93^{\Uparrow}$
DEC-apx	$0.95^{\Uparrow}$	$0.95^{\Uparrow}$	$0.94^{\Uparrow}$	$0.95^{\Uparrow}$	$0.92^{\Uparrow}$	$0.95^{\Uparrow}$	$0.44^{\Uparrow}$	$0.39^{\Uparrow}$	$0.43^{\Uparrow}$	$0.76^{\Uparrow}$
DEC-gapx	$0.95^{\Uparrow}$	$0.95^{\Uparrow}$	$0.95^{\Uparrow}$	$0.95^{\Uparrow}$	$0.80^{\Uparrow}$	$0.92^{\Uparrow}$	$0.46^{\Uparrow}$	$0.42^{\Uparrow}$	$\mathbf{0.35}^{\Downarrow}$	$\mathbf{0.23}^{\Downarrow}$
ours										
FedBNR	**0.03**	0.06	0.40	0.48	**0.08**	**0.10**	**0.19**	0.14	0.41	0.56
FedBNR-KD	**0.03**	$\mathbf{0.05}^{\downarrow}$	$\mathbf{0.25}^{\Downarrow}$	$\mathbf{0.30}^{\Downarrow}$	$0.12^{\Uparrow}$	$0.17^{\Uparrow}$	**0.19**	**0.12**	$0.47^{\Uparrow}$	0.55

Table A.8 Brier score (BRI) reported. Personalization setting. $\Uparrow$ and $\uparrow$ denote significantly worse results with $p < 0.01$ and $p < 0.05$ respectively; $\Downarrow$ and $\downarrow$ denote significantly better results similarly.

	Skillcraft		SML		Parkinsons		Bike		CCPP	
#clients	10	100	10	100	10	100	10	100	10	100
pFedGP	$0.30^{\Uparrow}$	$0.27^{\Uparrow}$	$0.31^{\Uparrow}$	$0.31^{\Uparrow}$	$0.31^{\Uparrow}$	$0.31^{\Uparrow}$	$0.30^{\Uparrow}$	$0.29^{\Uparrow}$	$0.33^{\Uparrow}$	$0.33^{\Uparrow}$
pxADMM	$0.30^{\Uparrow}$	$0.27^{\Uparrow}$	$0.33^{\Uparrow}$	$0.31^{\Uparrow}$	$0.33^{\Uparrow}$	$0.31^{\Uparrow}$	$0.22^{\Uparrow}$	$0.23^{\Uparrow}$	$0.33^{\Uparrow}$	$0.33^{\Uparrow}$
DEC-apx	$0.33^{\Uparrow}$	$0.33^{\Uparrow}$	$0.33^{\Uparrow}$	$0.33^{\Uparrow}$	$0.33^{\Uparrow}$	$0.33^{\Uparrow}$	$0.20^{\Uparrow}$	$\mathbf{0.17}^{\Downarrow}$	0.24	0.27
DEC-gapx	$0.33^{\Uparrow}$	$0.33^{\Uparrow}$	$0.33^{\Uparrow}$	$0.33^{\Uparrow}$	$0.30^{\Uparrow}$	$0.33^{\Uparrow}$	$0.20^{\Uparrow}$	$0.18^{\Downarrow}$	$\mathbf{0.16}^{\Downarrow}$	$\mathbf{0.16}^{\Downarrow}$
ours										
FedBNR	**0.17**	0.18	0.24	0.25	**0.18**	**0.19**	**0.19**	0.20	0.24	0.27
FedBNR-KD	**0.17**	$\mathbf{0.18}^{\downarrow}$	$\mathbf{0.21}^{\Downarrow}$	$\mathbf{0.22}^{\Downarrow}$	$0.19^{\Uparrow}$	$0.20^{\Uparrow}$	**0.19**	$0.19^{\Downarrow}$	$0.25^{\Uparrow}$	0.27

References

Achituve, I., Shamsian, A., Navon, A., Chechik, G., Fetaya, E.: Personalized federated learning with Gaussian processes. Advances in Neural Information Processing Systems **34** (2021)

Begoli, E., Bhattacharya, T., Kusnezov, D.: The need for uncertainty quantification in machine-assisted medical decision making. Nature Machine Intelligence **1**(1), 20–23 (2019)

Boenisch, F., Dziedzic, A., Schuster, R., Shamsabadi, A.S., Shumailov, I., Papernot, N.: When the curious abandon honesty: Federated learning is not private. arXiv preprint arXiv:2112.02918 (2021)

Byrd, D., Polychroniadou, A.: Differentially private secure multi-party computation for federated learning in financial applications. In: Proceedings of the First ACM International Conference on AI in Finance, pp. 1–9 (2020)

Cao, Y., Fleet, D.J.: Generalized product of experts for automatic and principled fusion of Gaussian process predictions. arXiv preprint arXiv:1410.7827 (2014)

Calandra, R., Peters, J., Rasmussen, C.E., Deisenroth, M.P.: Manifold Gaussian processes for regression. In: 2016 International Joint Conference on Neural Networks (IJCNN), pp. 3338–3345 (2016). IEEE

Cuturi, M.: Positive definite kernels in machine learning. arXiv preprint arXiv:0911.5367 (2009)

Damianou, A., Lawrence, N.D.: Deep gaussian processes. In: Artificial Intelligence and Statistics, pp. 207–215 (2013). PMLR

Deisenroth, M., Ng, J.W.: Distributed gaussian processes. In: International Conference on Machine Learning, pp. 1481–1490 (2015). PMLR

Danquah, B., Riedmaier, S., Rühm, J., Kalt, S., Lienkamp, M.: Statistical model verification and validation concept in automotive vehicle design. Procedia Cirp **91**, 261–270 (2020)

Fenton, L.: The sum of log-normal probability distributions in scatter transmission systems. IRE Transactions on communications systems **8**(1), 57–67 (1960)

Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: International Conference on Machine Learning, pp. 1050–1059 (2016). PMLR

Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y.W., Rezende, D., Eslami, S.A.: Conditional neural processes. In: International Conference on Machine Learning, pp. 1704–1713 (2018). PMLR

Gawlikowski, J., Tassi, C.R.N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., Triebel, R., Jung, P., Roscher, R., Shahzad, M., Yang, W., Bamler, R., Zhu, X.X.: A Survey of Uncertainty in Deep Neural Networks (2022)

Gal, Y., Van Der Wilk, M., Rasmussen, C.E.: Distributed variational inference in sparse Gaussian process regression and latent variable models. Advances in neural information processing systems **27** (2014)

Hensman, J., Fusi, N., Lawrence, N.D.: Gaussian processes for big data. arXiv preprint arXiv:1309.6835 (2013)

Hoang, T.N., Hoang, Q.M., Low, B.K.H.: A distributed variational inference framework for unifying parallel sparse Gaussian process regression models. In: International Conference on Machine Learning, pp. 382–391 (2016). PMLR

He, B., Ozay, M.: Feature kernel distillation. In: International Conference on Learning Representations (2022)

Hoeffding, W.: Probability inequalities for sums of bounded random variables. The collected works of Wassily Hoeffding, 409–426 (1994)

Hinton, G.E., Salakhutdinov, R.R.: Using deep belief nets to learn covariance kernels for Gaussian processes. Advances in neural information processing systems **20** (2007)

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

Iyer, A., Zhang, Y., Prasad, A., Tao, S., Wang, Y., Schadler, L., Brinson, L.C., Chen, W.: Data-centric mixed-variable bayesian optimization for materials design. In: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, vol. 59186, pp. 02–03066 (2019). American Society of Mechanical Engineers

Iyer, A., Zhang, Y., Prasad, A., Gupta, P., Tao, S., Wang, Y., Prabhune, P., Schadler, L.S., Brinson, L.C., Chen, W.: Data centric nanocomposites design via mixed-variable bayesian optimization. Molecular Systems Design & Engineering **5**(8), 1376–1390 (2020)

Kerkouche, R., Ács, G., Castelluccia, C., Genevès, P.: Privacy-preserving and bandwidth-efficient federated learning: An application to in-hospital mortality prediction. In: Proceedings of the Conference on Health, Inference, and Learning. CHIL '21, pp. 25–35. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3450439.3451859 . https://doi.org/10.1145/3450439.3451859

Kristiadi, A., Hein, M., Hennig, P.: Being Bayesian, Even Just a Bit, Fixes Overconfidence in ReLU Networks (2020)

Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S., Stich, S., Suresh, A.T.: Scaffold: Stochastic controlled averaging for federated learning. In: International Conference on Machine Learning, pp. 5132–5143 (2020). PMLR

Konečnỳ, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492 (2016)

Kontoudis, G.P., Stilwell, D.J.: Decentralized federated learning using gaussian processes. In: 2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS), pp. 1–7 (2023). IEEE

Kontoudis, G.P., Stilwell, D.J.: Scalable, federated gaussian process training for decentralized multi-agent systems. IEEE Access (2024)

Liu, H., Ong, Y.-S., Shen, X., Cai, J.: When Gaussian process meets big data: A review of scalable gps. IEEE transactions on neural networks and learning systems **31**(11), 4405–4423 (2020)

Li, Y., Rao, S., Hassaine, A., Ramakrishnan, R., Canoy, D., Salimi-Khorshidi, G., Mamouei, M., Lukasiewicz, T., Rahimi, K.: Deep bayesian gaussian processes for uncertainty estimation in electronic health records. Scientific reports **11**(1), 20685 (2021)

Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. Proceedings of Machine Learning and Systems **2**, 429–450 (2020)

Lederer, A., Umlauft, J., Hirche, S.: Uniform error bounds for gaussian process regression with application to safe control. Advances in Neural Information Processing Systems **32** (2019)

Li, Y., Zhou, Y., Jolfaei, A., Yu, D., Xu, G., Zheng, X.: Privacy-preserving federated learning framework based on chained secure multiparty computing. IEEE Internet of Things Journal **8**(8), 6178–6186 (2020)

McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282 (2017). PMLR

Mothukuri, V., Parizi, R.M., Pouriyeh, S., Huang, Y., Dehghantanha, A., Srivastava, G.: A survey on security and privacy of federated learning. Future Generation Computer Systems **115**, 619–640 (2021)

Mohri, M., Sivek, G., Suresh, A.T.: Agnostic federated learning. In: International Conference on Machine Learning, pp. 4615–4625 (2019). PMLR

Nickson, T., Gunter, T., Lloyd, C., Osborne, M.A., Roberts, S.: Blitzkriging: Kronecker-structured stochastic Gaussian processes. arXiv preprint arXiv:1510.07965 (2015)

Oliva, J.B., Dubey, A., Wilson, A.G., Póczos, B., Schneider, J., Xing, E.P.: Bayesian nonparametric kernel-learning. In: Artificial Intelligence and Statistics, pp. 1078–1086 (2016). PMLR

Ober, S.W., Rasmussen, C.E., Wilk, M.: The promises and pitfalls of deep kernel learning. In: Uncertainty in Artificial Intelligence, pp. 1206–1216 (2021). PMLR

Quinonero-Candela, J., Rasmussen, C.E.: A unifying view of sparse approximate Gaussian process regression. The Journal of Machine Learning Research **6**, 1939–1959 (2005)

Riedmaier, S., Danquah, B., Schick, B., Diermeyer, F.: Unified framework and survey for model verification, validation and uncertainty quantification. Archives of Computational Methods in Engineering **28**, 2655–2688 (2021)

Reisizadeh, A., Mokhtari, A., Hassani, H., Jadbabaie, A., Pedarsani, R.: Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In: International Conference on Artificial Intelligence and Statistics, pp. 2021–2031 (2020). PMLR

Riedmaier, S., Ponn, T., Ludwig, D., Schick, B., Diermeyer, F.: Survey on scenario-based safety assessment of automated vehicles. IEEE access **8**, 87456–87477 (2020)

Rahimi, A., Recht, B.: Random features for large-scale kernel machines. Advances in neural information processing systems **20** (2007)

Rudin, W.: Fourier Analysis on Groups. Courier Dover Publications, ??? (2017)

Saatçi, Y.: Scalable inference for structured Gaussian process models. PhD thesis, Citeseer (2012)

Shoham, N., Avidor, T., Keren, A., Israel, N., Benditkis, D., Mor-Yosef, L., Zeitak, I.: Overcoming forgetting in federated learning on non-iid data. arXiv preprint arXiv:1910.07796 (2019)

Smith, M.T., Álvarez, M.A., Lawrence, N.D.: Differentially private regression and classification with sparse Gaussian processes. J. Mach. Learn. Res. **22**, 188–1 (2021)

Smola, A., Bartlett, P.: Sparse greedy Gaussian process regression. Advances in neural information processing systems **13** (2000)

Sinha, A., Duchi, J.C.: Learning kernels with random features. Advances in Neural Information Processing Systems **29** (2016)

Snelson, E., Ghahramani, Z.: Sparse Gaussian processes using pseudo-inputs. Advances in neural information processing systems **18** (2005)

Schwaighofer, A., Tresp, V.: Transductive and inductive methods for approximate Gaussian process regression. Advances in neural information processing systems **15** (2002)

Seeger, M.W., Williams, C.K., Lawrence, N.D.: Fast forward selection to speed up sparse Gaussian process regression. In: International Workshop on Artificial Intelligence and Statistics, pp. 254–261 (2003). PMLR

Sattler, F., Wiedemann, S., Müller, K.-R., Samek, W.: Robust and communication-efficient federated learning from non-iid data. IEEE transactions on neural networks and learning systems **31**(9), 3400–3413 (2019)

Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., Zhou, Y.: A hybrid approach to privacy-preserving federated learning. In: Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, pp. 1–11 (2019)

Tran, G.-L., Bonilla, E.V., Cunningham, J., Michiardi, P., Filippone, M.: Calibrating deep convolutional Gaussian processes. In: The 22nd International Conference on Artificial Intelligence and Statistics, pp. 1554–1563 (2019). PMLR

Thompson, J.J., Blair, M.R., Chen, L., Henrey, A.J.: Video game telemetry as a critical tool in the study of complex skill learning. PloS one **8**(9), 75129 (2013)

Triastcyn, A., Faltings, B.: Federated learning with bayesian differential privacy. In: 2019 IEEE International Conference on Big Data (Big Data), pp. 2587–2596 (2019). IEEE

Titsias, M.: Variational learning of inducing variables in sparse Gaussian processes. In: Artificial Intelligence and Statistics, pp. 567–574 (2009). PMLR

Tsanas, A., Little, M., McSharry, P., Ramig, L.: Accurate telemonitoring of parkinson's disease progression by non-invasive speech tests. Nature Precedings, 1–1 (2009)

Tang, M., Ning, X., Wang, Y., Wang, Y., Chen, Y.: Fedgp: Correlation-based active client selection strategy for heterogeneous federated learning. arXiv preprint arXiv:2103.13822 (2021)

Tüfekci, P.: Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. International Journal of Electrical Power & Energy Systems **60**, 126–140 (2014)

VE, S., Cho, Y.: A rule-based model for seoul bike sharing demand prediction using

weather data. European Journal of Remote Sensing **53**(sup1), 166–183 (2020)

Verrelst, J., Rivera, J.P., Moreno, J., Camps-Valls, G.: Gaussian processes uncertainty estimates in experimental sentinel-2 lai and leaf chlorophyll content retrieval. ISPRS journal of photogrammetry and remote sensing **86**, 157–167 (2013)

Wilson, A.G., Dann, C., Nickisch, H.: Thoughts on massively scalable Gaussian processes. arXiv preprint arXiv:1511.01870 (2015)

Wilson, A.G., Hu, Z., Salakhutdinov, R.R., Xing, E.P.: Stochastic variational deep kernel learning. Advances in Neural Information Processing Systems **29**, 2586–2594 (2016)

Wilson, A.G., Hu, Z., Salakhutdinov, R., Xing, E.P.: Deep kernel learning. In: Artificial Intelligence and Statistics, pp. 370–378 (2016). PMLR

Wilcoxon, F.: Individual comparisons by ranking methods. In: Breakthroughs in Statistics, pp. 196–202. Springer, ??? (1992)

Wei, K., Li, J., Ding, M., Ma, C., Yang, H.H., Farokhi, F., Jin, S., Quek, T.Q., Poor, H.V.: Federated learning with differential privacy: Algorithms and performance analysis. IEEE Transactions on Information Forensics and Security **15**, 3454–3469 (2020)

Wilson, A., Nickisch, H.: Kernel interpolation for scalable structured Gaussian processes (kiss-gp). In: International Conference on Machine Learning, pp. 1775–1784 (2015). PMLR

Xia, X., Sivonxay, E., Helms, B.A., Blau, S.M., Chan, E.M.: Accelerating the design of multishell upconverting nanoparticles through bayesian optimization. Nano Letters **23**(23), 11129–11136 (2023)

Xie, A., Yin, F., Xu, Y., Ai, B., Chen, T., Cui, S.: Distributed Gaussian processes hyperparameter optimization for big data using proximal admm. IEEE Signal Processing Letters **26**(8), 1197–1201 (2019)

Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, N.: Statistical model aggregation via parameter matching. Advances in Neural Information Processing Systems **32** (2019)

Yin, F., Lin, Z., Kong, Q., Xu, Y., Li, D., Theodoridis, S., Cui, S.R.: Fedloc: Federated learning framework for data-driven cooperative localization and location data processing. IEEE Open Journal of Signal Processing **1**, 187–215 (2020)

Zamora-Martinez, F., Romeu, P., Botella-Rocamora, P., Pardo, J.: On-line learning of indoor temperature forecasting models towards energy efficiency. Energy and Buildings **83**, 162–172 (2014)

Zhu, H., Xu, J., Liu, S., Jin, Y.: Federated learning on non-iid data: A survey. Neurocomputing **465**, 371–390 (2021)