Personalized Federated Learning with Gaussian Processes

Anonymous Author(s)
Affiliation
Address
email

Abstract

Federated learning aims to learn a global model that performs well on client devices with limited cross-client communication. Personalized federated learning (PFL) further extends this setup to handle data heterogeneity between clients by learning personalized models. A key challenge in this setting, is to learn effectively across clients, even though each client has its unique data, which is often limited in size. Here we put forward a solution to PFL that is based on Gaussian processes (GPs) with deep kernel learning, which we call *pFedGP*. GP models are highly expressive, and thanks to their Bayesian nature, they work well in the low data regime. However, applying GPs to PFL raises multiple challenges. Mainly, GPs performance depends heavily on access to a good kernel function, and learning a kernel requires a large training set. To address this, we propose learning a single kernel function, parameterized by a neural network, shared by all clients using a personal GP classifier for each client. We further extend pFedGP to include inducing points, using two novel methods. We derive a PAC-Bayes generalization bound on novel clients and empirically show that it gives non-vacuous guarantees. We perform extensive experiments on standard PFL benchmarks as well as a new setup of learning under input noise. We find that pFedGP significantly outperforms baseline methods while achieving well-calibrated predictions.

1 Introduction

In recent years, there is a growing interest in applying learning in decentralized systems under the setup of federated learning (FL) [30, 44, 58]. In FL, a server node stores a global model and connects to multiple end-devices (“clients"), which have private data that cannot be shared. The goal is to learn the global model in a communication-efficient manner. However, learning a single shared model across all clients may perform poorly when the data distribution varies significantly across clients. *Personalized Federated Learning* (PFL) [59] addresses this challenge by jointly learning a personalized model for each client. While significant progress had been made in recent years, leading approaches still struggle in realistic scenarios. First, when the amount of data per client is limited, even though this is one of the original motivations behind federated learning [4, 44, 63]. Second, when the *input* distribution shifts between clients, which is often the case, as clients use different devices and sensors. Last, when we require well-calibrated predictions, which is an important demand from medical and other safety-critical applications.

Here, we show how *Gaussian Processes* (GPs) with deep kernel learning (DKL) [71] is an effective alternative for handling these challenges. GPs have good predictive performance in a wide range of dataset sizes [2, 72], they are robust to input noise [66], can adapt to shifts in the data distribution [41], and provide well-calibrated predictions [60]. While regression tasks are more natural for GPs, we focus on classification tasks for consistency with common benchmarks and learning procedures in the field; however, this approach is also applicable to regression tasks.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

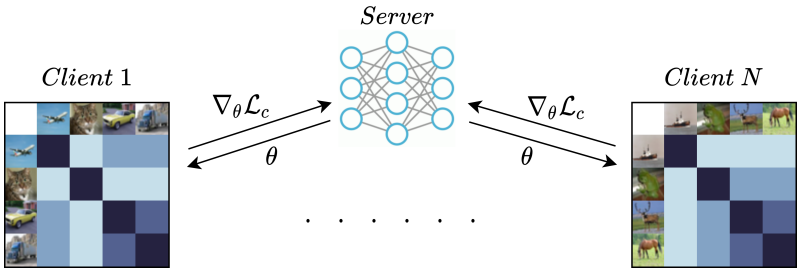

Figure 1: pFedGP - learning a shared deep kernel function with client-specific GP models. Each client stores private data, possibly from a different distribution. The data is first mapped to an embedding space with a shared neural network across all clients. Then, using common kernels a GP is applied to the data of the client for model learning and inference. We illustrate the per client kernel matrix $k_\theta(\mathbf{x}_i, \mathbf{x}_j)$. Bold cells indicate a stronger covariance.

Consider a naive approach that fits a separate GP classifier to each client based on its own data. Its performance heavily depends on the quality of the kernel, and standard kernels tend to work poorly in domains such as images. A popular solution to this problem is deep kernel learning [71] (DKL), where a kernel is applied to the learned features outputted by a neural network (NN). Unfortunately, GPs with DKL can strongly overfit, often even worse than standard NNs [49], and thus negate the main benefit of using a GP. We solve this issue by jointly learning a shared kernel function across clients. As the kernel captures similarities between inputs, a single kernel should work well across clients, while using a separate GP per client will give the required flexibility for personalization.

We adapt a GP classifier recently proposed by [2], which uses the Pólya-Gamma augmentation [50] in a tree-structure model, to the federated setting. This method, termed GP-Tree, showed good performance on both large and small datasets. We further extend our model by tailoring two inducing points (IPs) methods [61] to our setting. The first shares the inducing points between clients to help generalization in the low data regime and, unlike common inducing points methods, does not reduce the computational costs. The second inducing points method does focus on reducing the computational cost to make our approach scalable and work on low-resource clients. Finally, we adjust previous PAC-Bayes generalization bounds for GPs [53, 56] to include the Pólya-Gamma augmentation scheme. These bounds are suitable for cases where the kernel is not learned, such as when new clients arrive after the shared NN was already learned.

This paper makes the following contributions: (i) Introduce GPs as a natural fit to PFL setups; (ii) Develop two IPs methods to enhance GP classifiers that use the Pólya-Gamma augmentation scheme and extend our classifier with them; (iii) Derive a PAC-Bayes generalization bound on novel clients and show empirically that it gives useful guarantees; (iv) Achieve state-of-the-art results in a wide array of experiments, improving accuracy by up to 21%.

2 Related work

Federated learning (FL). In FL, clients collaboratively solve a learning task while preserving data privacy and maintaining communication efficiency [1, 27, 35, 44, 47, 73, 76]. FedAvg [44], is an early but effective FL approach that updates models locally and averages them into a global model. Several optimization methods have been proposed for improving convergence in FL [34, 38, 62, 68]. Other approaches focus on preserving client privacy [3, 15, 45, 81], improving robustness to statistical diversity [21, 22, 25, 28, 78, 79], and reducing communication cost [9, 54]. These methods aim to learn a global model across clients, limiting their ability to deal with heterogeneous datasets.

Personalized federated learning (PFL). To overcome client heterogeneity, PFL aims to introduce some personalization for each client in the federation [32, 64]. Recent methods include adapting multitask learning [13, 59], meta-learning approaches [5, 16, 17, 36, 80], and model mixing, where clients learn a mixture of the global and local models [4, 12, 22, 37]. Other approaches utilize different regularization schemes to enforce soft parameter sharing [26, 63]. Personalization in FL has also been explored through clustering approaches in which similar clients within the federation have

a greater effect on one another [42, 77]. Recently, [57] proposed learning a central hypernetworks that acts on client representation vectors for generating personalized models.

Bayesian FL. Some studies proposed a Bayesian treatment to the FL setup. [7, 8] proposed to use variational inference with Bayesian neural networks. In [67, 75] a matching algorithm based on the Beta-Bernoulli process was used between local models to construct a global model. [74] proposed a FL framework possibly with a global GP model in regression tasks and without DKL. We, on the other hand, focus on classification tasks and share information between clients via the learned kernel. [65] offered a client selection strategy with GPs. In [29] an approach based on stein-variational gradient descent was suggested. This method does not scale beyond small-sized networks. [40] proposed a multivariate Gaussian product mechanism to aggregate local models. As we will show, this method is less suited for the PFL learning setup.

Gaussian process classification. Unlike regression tasks, in classification the likelihood is not a Gaussian which forces us to use approximations [52]. Classic approaches include the Laplace approximation [70], expectation-propagation [46], and least squares [55]. Recently, several methods were proposed based on the Pólya-Gamma augmentation [50] for modeling multinomial distributions [39], GP classification [18, 19, 69], few-shot learning [60], and incremental learning [2]. Here we build on the last approach. Classification is commonly done with variational inference techniques [24], here we wish to exploit the conjugacy of the model to take Gibbs samples from the posterior. This approach yields well calibrated [60] and more accurate models [2].

3 Gaussian processes background

We first provide a brief introduction to Gaussian Processes (GPs). Detailed explanations are deferred to the Appendix. We denote scalars with lower-case letters (e.g., x), vectors with bold lower-case letters (e.g., $\mathbf{x}$), and matrices with bold capital letters (e.g., $\mathbf{X}$). In general, $\mathbf{y} = [y_1, ..., y_N]^T$ denotes the vector of labels, and $\mathbf{X} \in \mathbb{R}^{N \times d}$ the design matrix with N data points whose i^{th} row is $\mathbf{x}_i$.

Gaussian processes. GPs map input points to target output values via a random latent function f. f is assumed to follow a Gaussian process prior $f \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, where the evaluation vector of f on $\mathbf{X}$, $\mathbf{f} = [f(\mathbf{x}_1), ..., f(\mathbf{x}_N)]^T$, has a Gaussian distribution $\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{K})$, with means $\boldsymbol{\mu}_i = m(\mathbf{x}_i)$ and covariance $\boldsymbol{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. The mean $m(\mathbf{x})$ is often set to be the constant zero function, and the kernel $k(\mathbf{x}, \mathbf{x}')$ is a positive semi-definite function. The target values are assumed to be independent when conditioned on $\mathbf{f}$. For Gaussian process regression the likelihood is Gaussian, $p(y|f) = \mathcal{N}(f, \sigma^2)$. Therefore, the posterior $p(f|y, \mathbf{X})$ is also Gaussian, and both the marginal and the predictive distributions have known analytic expressions. This is one of the main motivations behind using GPs, since most other Bayesian models have intractable inference.

Unfortunately, for Gaussian process classification (GPC), the likelihood $p(y|f)$ is no longer Gaussian, and the posterior has no known closed-form expression. One approach for applying GPs to binary classification tasks is the Pólya-Gamma augmentation [50]. Using this approach, we can augment the GP distribution with a random variable ω for each data point, with $\boldsymbol{\omega}$ denoting the vector of all ω's. This distribution has a useful property: $p(\mathbf{f}|\mathbf{y}, \mathbf{X}, \boldsymbol{\omega})$ is Gaussian and $p(\boldsymbol{\omega}|\mathbf{y}, \mathbf{X}, \mathbf{f})$ has the Pólya-Gamma distribution [50] both with known parameters. This property allows us to use Gibbs sampling to efficiently sample from the posterior $p(\mathbf{f}, \boldsymbol{\omega}|\mathbf{y}, \mathbf{X})$ for training and inference. Full equations and further details on the Pólya-Gamma augmentation scheme are given in the Appendix.

Deep kernel learning (DKL). The quality of the GP prediction heavily depends on the kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$. For many data modalities, such as images, common kernels are not a good measure of semantic similarity. Therefore, in [71] the authors propose to use a standard kernel over features outputed by a NN, e.g. $k_\theta(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{||g_\theta(\mathbf{x}_i) - g_\theta(\mathbf{x}_i)||^2}{2\ell^2}\right)$. In regression tasks, it is possible to directly backpropagate through the GP inference as it is given in closed-form. However, since in our case we do classification with the Pólya-Gamma augmentation, we can only obtain stochastic gradients of the kernel [60]. This may be done with Fisher's identity [14].

Inducing points (IPs). GPs require storing and inverting a kernel matrix on the entire training set, limiting its scalability. A common solution to this problem is to use inducing points [51, 61]. The key idea is to learn $M \ll N$ inducing (pseudo) inputs and apply the GP inference formulas on them instead. This reduces the computational cost as now we need to invert an $M \times M$ matrix only.

GP-Tree. We build on GP-tree [2], a recent GP classifier that was shown to scale well with dataset size and the number of classes. GP-tree builds a binary class-hierarchy tree which is used to turn the multiclass classification problem into a sequence of binary decisions along the tree nodes. Each node in the tree fits a binary GP classifier based on the Pólya-Gamma augmentation scheme and the data associated to that node. The tree is constructed by computing a prototype for each class and then performing divisive hierarchical clustering on these prototypes to get the desired hierarchical tree. While it was superior to other leading GPC methods, we note here that it may still struggle in problems with vast amount of classes. This limitation may be reflected in our method as well when handling such cases. Further details on this method are given in the Appendix.

4 pFedGP: federated learning with Gaussian processes

Now we describe our approach for applying personalized federated learning (PFL) with Gaussian processes. First, we extend GP-Tree to the FL setup and show how to use Gibbs sampling to learn the NN parameters. Then, we present two alternatives for this method that use inducing points. The first is for extremely limited-size datasets, while the second allows controlling the computational resources. We name our method *pFedGP*. An illustration of our method is given in Figure 1.

4.1 A full GP model

The training procedure follows the standard protocol in this field [4, 37, 44]. We assume the existence of a server that holds the shared parameters (a NN). Let C denote the set of clients. For each client $c \in C$ we denote by D_c its local dataset of size N_c. At each training iteration the model is sent to $S \subseteq \{1, ..., |C|\}$ clients to perform kernel learning [71]. Each client $c \in S$ updates its copy of the global model and then sends the updated model to the server. The server then averages over the updates to obtain a new global model.

At each client c, we perform kernel learning in the following manner. We first compute the feature representation of the data samples associated with that client using the shared network. Then we build a hierarchical classification tree from data associated with that client as discussed in Section 3 (and the Appendix). We note that here, unlike in [2], we re-build the tree using the most recent features at each iteration. Of the two approaches proposed in [2] we use the Gibbs sampling approach, as it allows for flexibility in building the tree and performs better when not prohibitive by computational limitations. Here we extend this method to apply model learning with this approach. The two common objectives for learning the network parameters θ are maximizing the marginal likelihood or the predictive likelihood. For notational convenience, we omit the subscript c here from $\mathbf{X}$ and $\mathbf{y}$.

We denote by $\mathbf{X}_v$ the data associated with node v, i.e., the data points which have v on the path from the root node to their class leaf node. We denote by $\mathbf{y}_v$ the binary label of these points on the node, i.e., does their path go left or right on this node, and by $\boldsymbol{\omega}_v$ the Pólya-Gamma augmentation for node v. The marginal likelihood term for the full hierarchical classification tree is the sum of the separate marginal likelihood terms of all the nodes v in the tree:

$$\mathcal{L}_c^{ML}(\theta; D_c) = \sum_v \log\ p_\theta(\mathbf{y}_v|\mathbf{X}_v) = \sum_v \log \int p_\theta(\mathbf{y}_v|\boldsymbol{\omega}_v, \mathbf{X}_v) p(\boldsymbol{\omega}_v) d\boldsymbol{\omega}_v. \tag{1}$$

Similar to [60] we use an approximate-gradient estimator based on Fisher's identity [14]:

$$\nabla_\theta \mathcal{L}_c^{ML}(\theta; D_c) \approx \sum_v \frac{1}{L} \sum_{l=1}^{L} \nabla_\theta log\ p_\theta(\mathbf{y}_v|\boldsymbol{\omega}_v^{(l)}, \mathbf{X}_v). \tag{2}$$

Here, $\boldsymbol{\omega}_v^{(1)}, ..., \boldsymbol{\omega}_v^{(L)}$ are samples from the posterior at node v, and due to the Pólya-Gamma augmentation $p_\theta(\mathbf{y}_v|\boldsymbol{\omega}_v^{(l)}, \mathbf{X}_v)$ is proportional to a Gaussian density, which is given in the Appendix.

To use the predictive likelihood as an objective, in each training iteration, after building the tree model, at each node we randomly draw a portion from the (node) training data and use it to predict the class label for the remaining part. We denote with $\mathbf{X}_v$ and $\mathbf{y}_v$ the training portion, $\mathbf{x}_v^*$ and y_v^* the input and the label of the point we are predicting, and P^{y^*} the path from the root node to the y^* leaf node (i.e., the original class). Here we also take advantage of the independence between nodes to

maximize the likelihood per node individually. The predictive likelihood for a single data point:

$$\mathcal{L}_c^{PL}(\theta; \mathbf{x}^*, y^*) = \sum_{v \in P^{y^*}} \log\ p_\theta(y_v^* | \mathbf{x}_v^*, \mathbf{y}_v, \mathbf{X}_v) = \sum_{v \in P^{y^*}} \log \int p_\theta(y_v^* | \boldsymbol{\omega}_v, \mathbf{x}_v^*, \mathbf{y}_v, \mathbf{X}_v) p(\boldsymbol{\omega}_v) d\boldsymbol{\omega}_v. \tag{3}$$

Here, as well, we use an approximate-gradient estimator based on posterior samples of $\boldsymbol{\omega}$:

$$\nabla_\theta \mathcal{L}_c^{PL}(\theta; \mathbf{x}^*, y^*) \approx \sum_{v \in P^{y^*}} \frac{1}{L} \sum_{l=1}^{L} \nabla_\theta log\ p_\theta(y_v^* | \boldsymbol{\omega}_v^{(l)}, \mathbf{x}_v^*, \mathbf{y}_v, \mathbf{X}_v). \tag{4}$$

Where, $p_\theta(y_v^* | \boldsymbol{\omega}_v^{(l)}, \mathbf{x}_v^*, \mathbf{y}_v, \mathbf{X}_v) = \int p(y_v^* | f^*) p_\theta(f^* | \boldsymbol{\omega}_v^{(l)}, \mathbf{x}_v^*, \mathbf{y}_v, \mathbf{X}_v) df^*$ does not have an analytical expression, but $p_\theta(f^* | \boldsymbol{\omega}_v^{(l)}, \mathbf{x}_v^*, \mathbf{y}_v, \mathbf{X}_v) = \int p_\theta(f^* | \mathbf{f}, \mathbf{x}_v^*, \mathbf{X}_v) p_\theta(\mathbf{f} | \boldsymbol{\omega}_v^{(l)}, \mathbf{y}_v, \mathbf{X}_v) df^*$ is Gaussian with known parameters. We then compute the likelihood by performing Gauss-Hermite integration over f^*. See exact expression in the Appendix.

4.2 Augmenting the model with inducing points: sample efficiency

The GP model described in Section 4.1 works well in most situations. However, when the number of data points per client is small, performance naturally degrades. To increase information sharing between clients and improve the per-client performance, we suggest augmenting the model with global inducing points (IPs) shared across clients. When sending the model from the server to a client, we also send the inducing inputs and their labels. To streamline optimization and reduce the communication burden, we define the IPs in the feature space of the last embedding layer of the shared NN. Therefore, usually, their size will be negligible compared to the network size.

We denote by $\bar{\mathbf{X}}$ the learned inducing inputs and by $\bar{\mathbf{y}}$ their fixed class labels. During training, we optimize the predictive likelihood (Equation 3) of client data given the inducing points. More formally, we compute $p_\theta(\mathbf{f} | \bar{\boldsymbol{\omega}}, \bar{\mathbf{y}}, \bar{\mathbf{X}}, \mathbf{X}) = \int p_\theta(\mathbf{f} | \bar{\mathbf{f}}, \bar{\mathbf{X}}, \mathbf{X}) p_\theta(\bar{\mathbf{f}} | \bar{\boldsymbol{\omega}}, \bar{\mathbf{y}}, \bar{\mathbf{X}}) d\bar{\mathbf{f}}$ using its analytical expression and compute the likelihood of $\mathbf{y}$ using Gauss-Hermite integration. At test time, to make full use of the training data, we combine the inducing points with the training data and use both to make predictions. We note that with just using the inducing points at test time, the model performs remarkably well, despite having almost no personalization component. See the Appendix for a further discussion.

One issue that arises when introducing IPs is that they can distort the true class distribution and make the classifier more likely to predict a low-probability class. We address this issue by adjusting the output distribution. In general, let $p(y, \mathbf{x})$ and $q(y, \mathbf{x})$ be two distributions that differ only in the class probabilities, i.e. $p(\mathbf{x}|y) = q(\mathbf{x}|y)$, we get that the predictive distribution follows

$$\frac{q(y^*|\mathbf{x}^*)}{p(y^*|\mathbf{x}^*)} \propto \frac{q(\mathbf{x}^*|y^*) q(y^*)}{p(\mathbf{x}^*|y^*) p(y^*)} \implies q(y^*|\mathbf{x}^*) \propto \frac{q(y^*)}{p(y^*)} p(y^*|\mathbf{x}^*). \tag{5}$$

We use this to correct the GP predictions to the original class ratios. We found in our experiments that this correction generally improves the classifier performance for unbalanced data.

4.3 Augmenting the model with inducing points: computational efficiency

Learning the full GP model described in Section 4.1 requires inverting a matrix of size N_c in the worst case (at the root node), which has $\mathcal{O}(N_c^3)$ run-time complexity and $\mathcal{O}(N_c^2)$ memory complexity. Therefore, we also propose an additional procedure for using the inducing points method to allow reduced complexity in low resource environments and scalability to larger data sizes per client.

This variant is based on the fully independent training conditional (FITC) method [61]. The key idea is to cast all the dependence on the inducing points and assume independence between the latent function values given the inducing points. Here for conciseness, we omit the subscripts denoting the client and the tree node. However, all quantities and data points are those that belong to a specific client and tree node. Let $\bar{\mathbf{X}} \in \mathbb{R}^{M \times d}$ denote the pseudo-inputs (defined in the embedding space of the last layer of the NN), and $\bar{\mathbf{f}} \in \mathbb{R}^M$ the corresponding latent function values. Here as well, the inducing inputs are defined globally at the server level. We assume the following GP prior $p(\mathbf{f}, \bar{\mathbf{f}}) = \mathcal{N}\Big(\mathbf{0}, \begin{bmatrix} \mathbf{K}_{NN} & \mathbf{K}_{NM} \\ \mathbf{K}_{MN} & \mathbf{K}_{MM} \end{bmatrix}\Big)$, where $\mathbf{K}_{MM}$ is the kernel between the inducing inputs, $\mathbf{K}_{NN}$

is a diagonal matrix between the actual training data, $\mathbf{K}_{NM}$ is the kernel between the data and the inducing inputs, and we placed a zero mean prior. The likelihood of the dataset when factoring the inducing variables and the Pólya-Gamma variables (one per training sample), and the posterior over $\mathbf{f}$, both have known analytical expressions. We can then obtain the posterior and marginal distributions by marginalizing over $\bar{\mathbf{f}}$. Here we will present the posterior and marginal distributions:

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \boldsymbol{\omega}, \bar{\mathbf{X}}) = \mathcal{N}(\mathbf{f}|\mathbf{K}_{NM}\mathbf{B}^{-1}\mathbf{K}_{MN}\boldsymbol{\Lambda}^{-1}\boldsymbol{\Omega}^{-1}\boldsymbol{\kappa}, \mathbf{K}_{NN} - \mathbf{K}_{NM}(\mathbf{K}_{MM}^{-1} - \mathbf{B}^{-1})\mathbf{K}_{MN}). \quad (6)$$

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\omega}, \bar{\mathbf{X}}) = \mathcal{N}(\boldsymbol{\Omega}^{-1}\boldsymbol{\kappa}|\mathbf{0}, \boldsymbol{\Lambda} + \mathbf{K}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{K}_{MN}). \quad (7)$$

where, $\boldsymbol{\Omega} = diag(\boldsymbol{\omega})$, $\kappa_j = y_j - 1/2$, $\boldsymbol{\Lambda} = \boldsymbol{\Omega}^{-1} + diag(\mathbf{K}_{NN} - \mathbf{K}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{K}_{MN})$ and $\mathbf{B} = \mathbf{K}_{MM} + \mathbf{K}_{MN}\boldsymbol{\Lambda}^{-1}\mathbf{K}_{NM}$. The important thing to note is that we only need to invert $M \times M$ or diagonal matrices. See full derivation in the appendix. During test time, in a similar manner, we use $\bar{\mathbf{f}}$ to get the posterior of $\mathbf{f}$ and use it to make the predictions.

Now we can use the marginal or predictive distributions similarly to our proposal in Section 4.1 to learn the shared NN parameters. The complexity of applying this procedure is reduced to $\mathcal{O}(M^2N_c + M^3)$ in run-time, and $\mathcal{O}(MN_c + M^2)$ in memory. While the (conditional) independence assumption between the latent function values may be restrictive, we found this method to be comparable with the full GP alternative in our experiments. Potentially, this can be attributed to the effect of sharing the inducing inputs among clients and the information that $\boldsymbol{\omega}$ stores on $\mathbf{f}$.

5 Generalization bound

It is reasonable to expect that after we learn the system, new clients will arrive. In such cases, we would like to use pFedGP without re-training the kernel function. Under this scenario, we can derive generalization bounds concerning only the GP classifier without taking into account the fixed neural network, using PAC-Bayes bound [43]. Having meaningful guarantees can be very important in safety-critical applications. The PAC-Bayes bound for GPC [56] (with the Gibbs risk):

Theorem 1. *Given i.i.d. samples $D_c = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_c}$ of size N_c drawn from any data distribution over $\mathcal{X} \times \{-1, 1\}$, a GP posterior Q, and a GP prior P, the following bound holds, where the probability is over random data samples:*

$$Pr_{D_c}\{R(Q) > R_{D_c}(Q) + KL_{ber}^{-1}(R_{D_c}(Q), \epsilon(\delta, n, P, Q))\} \leq \delta. \quad (8)$$

Here, we have,

$$R(Q) = \mathbb{E}_{(\mathbf{x}^*, y^*)}[Pr_{f^* \sim Q(f^*|\mathbf{x}^*, D_c)}\{sign\ f^* \neq y^*\}], \quad R_{D_c}(Q) = \frac{1}{N_c}\sum_{i=1}^{N_c} Pr_{f_i \sim Q(f_i|D_c)}\{sign\ f_i \neq y_i\}$$

$$\epsilon(\delta, N_c, P, Q) = \frac{1}{N_c}\Big(KL[Q \,||\, P] + \log\ \frac{N_c + 1}{\delta}\Big), \quad KL_{ber}^{-1}(q, \epsilon) = max_{p \in [0,1]} KL_{ber}[q \,||\, p] \leq \epsilon \quad (9)$$

An important observation in [56] is that the KL-divergence between the posterior and prior Gaussian processes is equivalent to the KL-divergence between the posterior and prior distribution of their values on the N_c training samples. While [56] assumed Q to be Gaussian, this observation still holds even without this assumption. However, when Q is no longer Gaussian, as is the case here, $KL[Q(\mathbf{f}) \,||\, P(\mathbf{f})]$ no longer has a closed-form expression. We can show that for the Pólya-Gamma augmentation:

$$KL[Q(\mathbf{f}) \,||\, P(\mathbf{f})] = \mathbb{E}_{Q(\boldsymbol{\omega})}\{KL[Q(\mathbf{f}|\boldsymbol{\omega})||P(\mathbf{f})]\} - MI[\mathbf{f}; \boldsymbol{\omega}]$$
$$= \mathbb{E}_{Q(\boldsymbol{\omega})}\{KL[Q(\mathbf{f}|\boldsymbol{\omega})||P(\mathbf{f})]\} + \mathbb{E}_{Q(\mathbf{f}, \boldsymbol{\omega})}\left[\log\ \frac{Q(\boldsymbol{\omega})}{Q(\boldsymbol{\omega}|\mathbf{f})}\right] \quad (10)$$

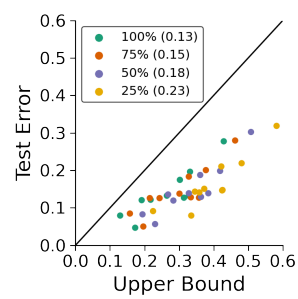

Figure 2: Test error vs an estimated upper bound over 10 clients with varying degrees of training set data sizes. In parenthesis - the average difference between the empirical and the test error.

where MI denotes the mutual information. Since $Q(\mathbf{f}|\boldsymbol{\omega})$ and $P(\mathbf{f})$ are Gaussian, the $KL[Q(\mathbf{f}|\boldsymbol{\omega})||P(\mathbf{f})]$ term has a close form expression so we only need to perform Monte-Carlo

approximation on the expectation on $\boldsymbol{\omega}$ on the first element. In the second expectation $Q(\boldsymbol{\omega})$ does not have a known expression. To estimate it, given $\{(\boldsymbol{\omega}_i, \mathbf{f}_i)\}_{i=1}^N$ samples, we use $Q(\boldsymbol{\omega}_i) \approx \frac{1}{N-1}\sum_{j\neq i} Q(\boldsymbol{\omega}_i|\mathbf{f}_j)$. Note that if the summation for i includes $\mathbf{f}_i$, it might result in a biased estimator. Further details on estimating $KL[Q(\mathbf{f}) \mid\mid P(\mathbf{f})]$ are in the Appendix.

To assess the quality of the bound, we partitioned the CIFAR-10 dataset to 100 clients. We trained a shared network using our full-GP variant on 90 clients and then recorded the generalization and test error on the remaining 10 clients four times, each with a different training set size. Figure 2 shows the estimation of the generalization error bound ($\delta = 0.01$) vs the actual error on the novel clients with the Gibbs classifier. First, we observe that indeed the bound is greater than the actual test error for all points and that it is not vacuous. There is a strong correlation between the actual error and the bound. Secondly, unlike worst-case bounds (e.g., VC-dimension), this bound depends on the actual data and not only the number of data points. Further results with the Bayes classifier are in the Appendix.

6 Experiments

We evaluate pFedGP against baseline methods on various learning setups. We present the result for the following model variants: *(i) pFedGP*, the full GP model (Section 4.1); *(ii) pFedGP-IP-data*, the model with IP described in Section 4.2; and *(iii) pFedGP-IP-compute*, the model with IP described in Section 4.3. For pFedGP and pFedGP-IP-compute, the results obtained by maximizing the predictive likelihood were similar to those obtained by maximizing the marginal likelihood, with a slight advantage to the former. Therefore, we present here the results only for the predictive alternative and defer the results of the marginal alternative to the Appendix. Additional experiments, full implementation details, ablation studies, and further analysis are also provided in the Appendix. Unless stated otherwise, we report the average and the standard error of the mean (SEM) over three random seeds of the federate accuracy defined as the average accuracy across all clients and samples.

Datasets. All methods were evaluated on CIFAR-10, CIFAR-100 [31], and CINIC-10 [10] datasets. CINIC-10 is more diverse since it combines images from CIFAR-10 and ImageNet [11].

Compared methods. We compared our method against the following baselines: *(1) Local*, pFedGP full model on each client with a private network and no collaboration with other clients; *(2) FedAvg* [44], a standard FL model with no personalization component; *(3) FOLA* [40], a Bayesian method that used a multivariate Gaussian product mechanism to aggregate local models; *(4) FedPer* [4], a PFL approach that learns a personal classifier for each client on top of a shared feature extractor; *(5) LG-FedAvg* [37], a PFL method that uses local feature extractor per client and global output layers; *(6) pFedMe* [63], a PFL method which adds a Moreau-envelopes loss term; *(7) FedU* [13], a recent multi-task learning approach for PFL that learns a model per client; *(8) pFedHN* [57], a recent PFL approach that uses a hypernetwork to generate client-specific networks.

Training protocol. We follow the training strategy proposed in [57]. We limit the training process to 1000 communication rounds, in each we sample five clients uniformly at random for model updates. The training procedure is different in the FOLA and pFedHN baselines, so we used an equivalent communication cost. In LG-FedAvg, we made an extra 200 communication rounds after a pre-training stage with the FedAvg model for 1000 communication rounds. In the local model, we performed 100 epochs of training for each client. In all experiments, we use a LeNet-based network [33] with two convolution and two fully connected layers. We tune the hyperparameters of all methods using a pre-allocated held-out validation set. Full experimental details are given in the Appendix.

6.1 Personalization through class probabilities

We evaluate all methods using a standard PFL benchmark where each client is assigned with different classes and training set size [57, 63]. We vary the total number of clients in the system from 50 to 500. For CIFAR-10/CIFAR-100/CINIC-10, each client is allocated with two/ten/four classes, respectively. Since the total number of samples kept fixed, the number of samples per client changes accordingly.

The results are presented in Table 1. They show that: (1) The performance of the *local* baseline is significantly impaired when the number of samples per client decreases, emphasizing the importance of federated learning in the presence of limited local data. (2) FedAvg and FOLA, which do not use personalized FL, perform poorly in this heterogeneous setup. (3) pFedGP outperforms or is on par with previous state-of-the-art approaches when local data is sufficient (e.g., 50 clients on all

Table 1: Test accuracy ($\pm$ SEM) over 50, 100, 500 clients for CIFAR-10, CIFAR-100, and CINIC-10. The *# samples/client* indicates the average number of training samples per client.

	CIFAR-10			CIFAR-100			CINIC-10		
# clients	50	100	500	50	100	500	50	100	500
# samples/client	800	400	80	800	400	80	1800	900	180
Local	86.2 ± 0.2	82.9 ± 0.4	74.8 ± 0.5	52.1 ± 0.2	45.6 ± 0.3	30.9 ± 0.2	61.1 ± 0.3	56.9 ± 0.7	46.4 ± 0.1
FedAvg [44]	56.4 ± 0.5	59.7 ± 0.5	54.0 ± 0.5	23.6 ± 0.2	24.0 ± 0.2	20.4 ± 0.0	45.6 ± 0.4	44.7 ± 0.5	45.7 ± 0.5
FOLA [40]	55.9 ± 3.3	52.1 ± 3.1	45.9 ± 0.3	25.5 ± 1.5	22.4 ± 1.3	18.7 ± 0.1	45.2 ± 0.3	43.4 ± 0.3	38.3 ± 0.2
FedPer [4]	83.8 ± 0.8	81.5 ± 0.5	76.8 ± 1.2	48.3 ± 0.6	43.6 ± 0.2	25.6 ± 0.3	70.6 ± 0.2	68.4 ± 0.5	$62.2 \pm .05$
LG-FedAvg [37]	87.9 ± 0.3	83.6 ± 0.7	64.7 ± 0.7	43.6 ± 0.2	37.5 ± 0.9	20.3 ± 0.5	59.5 ± 1.1	59.9 ± 2.1	52.5 ± 0.8
pFedMe [63]	86.4 ± 0.8	85.0 ± 0.3	80.3 ± 0.5	49.8 ± 0.5	47.7 ± 0.4	32.5 ± 0.8	69.9 ± 0.5	68.9 ± 0.7	58.8 ± 0.1
FedU [13]	80.6 ± 0.3	78.1 ± 0.5	65.6 ± 0.4	41.1 ± 0.2	36.0 ± 0.2	15.9 ± 0.4	59.3 ± 0.2	55.4 ± 0.6	41.6 ± 0.5
pFedHN [57]	$\mathbf{90.2 \pm 0.6}$	87.4 ± 0.2	83.2 ± 0.8	60.0 ± 1.0	52.3 ± 0.5	34.1 ± 0.1	70.4 ± 0.4	69.4 ± 0.5	$64.2 \pm .05$
Ours									
pFedGP-IP-data	88.6 ± 0.2	87.4 ± 0.2	86.9 ± 0.7	60.2 ± 0.3	58.5 ± 0.3	$\mathbf{55.7 \pm 0.4}$	69.8 ± 0.2	68.3 ± 0.6	67.6 ± 0.3
pFedGP-IP-compute	$\mathbf{89.9 \pm 0.6}$	$\mathbf{88.8 \pm 0.1}$	86.8 ± 0.4	61.2 ± 0.4	59.8 ± 0.3	49.2 ± 0.3	$\mathbf{72.0 \pm 0.3}$	$\mathbf{71.5 \pm 0.5}$	$\mathbf{68.2 \pm 0.2}$
pFedGP	89.2 ± 0.3	$\mathbf{88.8 \pm 0.2}$	$\mathbf{87.6 \pm 0.4}$	$\mathbf{63.3 \pm 0.1}$	$\mathbf{61.3 \pm 0.2}$	50.6 ± 0.2	$\mathbf{71.8 \pm 0.3}$	$\mathbf{71.3 \pm 0.4}$	$\mathbf{68.1 \pm 0.3}$

datasets). When the data size per node becomes limited, pFedGP achieves significant improvements over competing methods; note the 9% and 21% difference in CIFAR-100 over 100 and 500 clients, respectively. (4) pFedGP-IP-compute often achieves comparable results to pFedGP. (5) pFedGP-IP-data is especially helpful when few samples per class are available, e.g., CIFAR-100 with 500 clients. That last point is further illustrated by decoupling the effect of the *number of clients* from that of the *training set size*. In the appendix, we further experimented with fixing the number of clients and varying the number of training samples per class. From this experiment, we deduced that both factors (individually) contribute to pFedGP success.

A desired property from PFL classifiers is the ability to provide uncertainty estimation. For example, in decision support systems, such as in healthcare applications, the decision-maker should have an accurate estimation of the classifier confidence in the prediction. Here, we quantify the uncertainty through calibration. Figure 3 compares all methods both visually and using common metrics [6, 20, 48] on the CIFAR-100 dataset with 50 clients. Expected calibration error (ECE) measures the weighted average between the classifier confidence and accuracy. Maximum calibration error (MCE) takes the maximum instead of the average. Brier score (BRI) [6] measures the average squared error between the labels and the prediction probabilities. The figure shows that pFedGP classifiers are best calibrated across all metrics in almost all cases. We note that with temperature scaling, the calibration of the baseline methods can be improved [20]; however, choosing the right temperature requires optimization over a separate validation set, which our model does not need. Additional calibration results, including temperature scaling, are presented in the Appendix.

Table 2: Test accuracy ($\pm$ SEM) over 100 clients on noisy CIFAR-100 dataset. We also provide the relative accuracy decrease (%) w.r.t. the performance on the original CIFAR-100 data (see Table 1).

Method	Accuracy	Decrease (%)
FedPer [4]	28.1 ± 0.9	-35.6
LG-FedAvg [37]	26.9 ± 0.9	-28.3
pFedme [63]	33.2 ± 0.6	-30.4
FedU [13]	35.0 ± 0.2	**-2.8**
pFedHN [57]	38.9 ± 0.5	-25.7
Ours		
pFedGP-IP-data	45.0 ± 0.3	-23.1
pFedGP-IP-compute	$47.1 \pm .05$	-21.2
pFedGP	$\mathbf{49.5 \pm 0.1}$	-19.2

6.2 PFL with input noise

In real-world federated systems, the clients may employ different measurement devices for data collection (cameras, sensors, etc.), resulting in different input noise characteristics per client. Here, we investigate pFedGP performance in this type of personalization. To simulate that, we first defined 57 unique distributions of image corruption noise [23]. When partitioning the data to 100 clients, similarly to the protocol described in Section 6.1, we also sampled a noise model for each client. Then for each example in each client we sampled a corruption noise according to the noise model allocated to that client. The noisy CIFAR-100 dataset will be made publicly available for future research. Further details on the perturbations performed are given in the Appendix. The results are given in Table 2. We observe a significant gap in favor of the pFedGP variants compared to baseline methods. Note that using global inducing points is slightly less beneficial in this case since they are defined globally and therefore are not tied to a specific noise type as the real client data was.

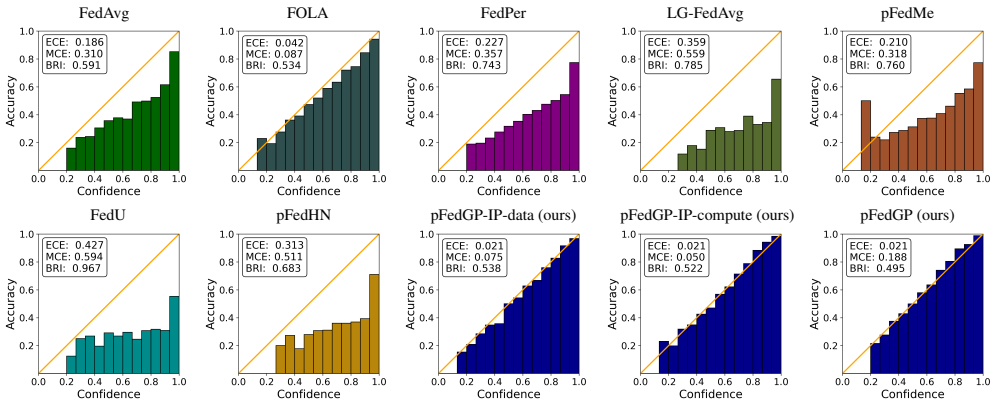

Figure 3: Reliability diagrams on CIFAR-100 with 50 clients. Each plot shows the expected & maximum calibration error (ECE & MCE) and the Brier Score (BRI). Lower is better. Diagonal indicates perfect calibration.

6.3 Generalization to out-of-distribution (OOD) novel clients

FL are dynamic systems. For example, novel clients may enter the system after the model was trained, possibly with a data distribution shift. Adapting to a new OOD client is both challenging and important for real-world FL systems. To evaluate pFedGP in this scenario, we followed the learning protocol proposed in [57]. We partitioned the CIFAR-10 dataset into two groups. The data in the first group was distributed between 90 clients for model training. The remaining data from the second group was distributed between an additional 10 clients that were excluded during training. Within each group, we set the class probabilities in each client by sampling from Dirichlet distribution with the same α parameter. For the training group, we set $\alpha = 0.1$, trained the shared model using these clients, and froze it. Then, at the second stage, we examined the models on the second group by varying $\alpha \in \{.1, .25, .5, .75, 1\}$, on the remaining 10 clients. As α moves away from 0.1 the distribution shift between the two groups increases, resulting in more challenging OOD clients. Figure 4 reports the generalization gap as a function of the Dirichlet parameter α. The generalization gap is computed by taking the difference between the average test accuracy on the (ten) novel clients and the average test accuracy on the (ninety) clients used for training. From the figure, we observe, here as well, that pFedGP achieves the best generalization performance for all values of α. Several baselines were excluded from the figure since they had a large generalization gap.

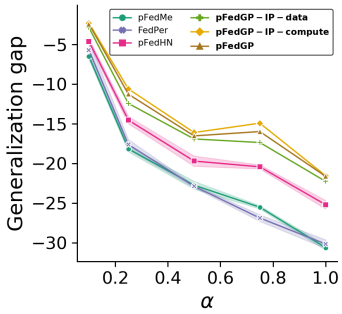

Figure 4: Generalization to novel clients on CIFAR-10 (lower is better).

7 Conclusion

In this study, we proposed pFedGP, a novel method for PFL. pFedGP learns a kernel function, parameterized by a NN, shared between all clients using a personal GP classifier on each client. We proposed three variants for pFedGP, and we derived PAC-Bayes generalization bound on novel clients. pFedGP provides well-calibrated predictions, generalizes well to OOD novel clients, and consistently outperforms competing methods.

Broader impact: Our method shares the standard communication procedure of FL approaches, where no private data is directly communicated across the different nodes in the system. This protocol does not explicitly guarantee that no private information can be inferred at this time. As we show, pFedGP is particularly useful for clients with little data, and for clients that have strongly different distribution. This has great potential to improve client personalization in real-world systems, and do better at handling less common data. The latter is of great interest for decision support systems in sensitive domains such as health care or legal.

References

[1] Sawsan Abdulrahman, Hanine Tout, Hakima Ould-Slimane, Azzam Mourad, Chamseddine Talhi, and Mohsen Guizani. A survey on federated learning: The journey from centralized to distributed on-site learning and beyond. *IEEE Internet of Things Journal*, 8(7):5476–5497, 2021.

[2] Idan Achituve, Aviv Navon, Yochai Yemini, Gal Chechik, and Ethan Fetaya. GP-Tree: A Gaussian process classifier for few-shot incremental learning. *arXiv preprint arXiv:2102.07868*, 2021.

[3] Naman Agarwal, Ananda Theertha Suresh, Felix Xinnan X Yu, Sanjiv Kumar, and Brendan McMahan. cpSGD: Communication-efficient and differentially-private distributed SGD. In *Advances in Neural Information Processing Systems*, pages 7564–7575, 2018.

[4] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

[5] Harkirat Singh Behl, Atılım Güneş Baydin, and Philip HS Torr. Alpha MAML: Adaptive model-agnostic meta-learning. *arXiv preprint arXiv:1905.07435*, 2019.

[6] Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

[7] Thang D Bui, Cuong V Nguyen, Siddharth Swaroop, and Richard E Turner. Partitioned variational inference: A unified framework encompassing federated and continual learning. *arXiv preprint arXiv:1811.11206*, 2018.

[8] Luca Corinzia, Ami Beuret, and Joachim M Buhmann. Variational federated multi-task learning. *arXiv preprint arXiv:1906.06268*, 2019.

[9] Xinyan Dai, Xiao Yan, Kaiwen Zhou, Han Yang, Kelvin KW Ng, James Cheng, and Yu Fan. Hyper-sphere quantization: Communication-efficient SGD for federated learning. *arXiv preprint arXiv:1911.04655*, 2019.

[10] Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. CINIC-10 is not Imagenet or CIFAR-10. *arXiv preprint arXiv:1810.03505*, 2018.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[12] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

[13] Canh T Dinh, Tung T Vu, Nguyen H Tran, Minh N Dao, and Hongyu Zhang. FedU: A unified framework for federated multi-task learning with Laplacian regularization. *arXiv preprint arXiv:2102.07148*, 2021.

[14] Randal Douc, Eric Moulines, and David Stoffer. *Nonlinear time series: Theory, methods and applications with R examples*. CRC press, 2014.

[15] John C Duchi, Michael I Jordan, and Martin J Wainwright. Privacy aware learning. *Journal of the ACM (JACM)*, 61(6):1–57, 2014.

[16] Alireza Fallah, Aryan Mokhtari, and A. Ozdaglar. Personalized federated learning: A meta-learning approach. *Advances in Neural Information Processing Systems*, 2020.

[17] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *International Conference on Artificial Intelligence and Statistics*, pages 1082–1092. PMLR, 2020.

[18] Théo Galy-Fajou, Florian Wenzel, Christian Donner, and Manfred Opper. Multi-class Gaussian process classification made conjugate: Efficient inference via data augmentation. In *Uncertainty in Artificial Intelligence*, pages 755–765. PMLR, 2020.

[19] Théo Galy-Fajou, Florian Wenzel, and Manfred Opper. Automated augmented conjugate inference for non-conjugate gaussian process models. In *International Conference on Artificial Intelligence and Statistics*, pages 3025–3035. PMLR, 2020.

[20] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

[21] Farzin Haddadpour and Mehrdad Mahdavi. On the convergence of local descent methods in federated learning. *arXiv preprint arXiv:1910.14425*, 2019.

[22] Filip Hanzely and Peter Richtárik. Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516*, 2020.

[23] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

[24] James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational Gaussian process classification. In *Artificial Intelligence and Statistics*, pages 351–360. PMLR, 2015.

[25] T. H. Hsu, Hang Qi, and M. Brown. Measuring the effects of non-identical data distribution for federated visual classification. *ArXiv*, abs/1909.06335, 2019.

[26] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on non-IID data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[27] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[28] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[29] Rahif Kassab and Osvaldo Simeone. Federated generalized Bayesian learning via distributed Stein variational gradient descent. *arXiv preprint arXiv:2009.06419*, 2020.

[30] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[31] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[32] V. Kulkarni, Milind Kulkarni, and A. Pant. Survey of personalization techniques for federated learning. *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pages 794–797, 2020.

[33] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[34] Tian Li, Anit Kumar Sahu, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. On the convergence of federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

[35] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[36] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-SGD: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

[37] Paul Pu Liang, Terrance Liu, Liu Ziyin, Nicholas B Allen, Randy P Auerbach, David Brent, Ruslan Salakhutdinov, and Louis-Philippe Morency. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523*, 2020.

[38] Tao Lin, Sebastian U Stich, Kumar Kshitij Patel, and Martin Jaggi. Don't use large mini-batches, use local SGD. In *International Conference on Learning Representations*, 2019.

[39] Scott W Linderman, Matthew J Johnson, and Ryan P Adams. Dependent multinomial models made easy: stick breaking with the Pólya-Gamma augmentation. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pages 3456–3464, 2015.

[40] Liangxi Liu and Feng Zheng. A Bayesian federated learning framework with multivariate gaussian product. *arXiv preprint arXiv:2102.01936*, 2021.

[41] Wesley Maddox, Shuai Tang, Pablo Moreno, Andrew Gordon Wilson, and Andreas Damianou. Fast adaptation with linearized neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 2737–2745. PMLR, 2021.

[42] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh. Three approaches for personalization with applications to federated learning. *ArXiv*, abs/2002.10619, 2020.

[43] David A McAllester. PAC-Bayesian stochastic model selection. *Machine Learning*, 51(1):5–21, 2003.

[44] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[45] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Representations*, 2018.

[46] Thomas Peter Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001.

[47] Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021.

[48] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using Bayesian binning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[49] Sebastian W. Ober, Carl E. Rasmussen, and Mark van der Wilk. The promises and pitfalls of deep kernel learning. *CoRR*, 2021.

[50] Nicholas G. Polson, James G. Scott, and Jesse Windle. Bayesian inference for logistic models using Pólya–Gamma latent variables. *Journal of the American Statistical Association*, pages 1339–1349, 2013.

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

[52] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[53] David Reeb, Andreas Doerr, Sebastian Gerwinn, and Barbara Rakitsch. Learning Gaussian processes by minimizing PAC-Bayesian generalization bounds. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 3341–3351, 2018.

[54] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. FedPAQ: A communication-efficient federated learning method with periodic averaging and quantization. In *International Conference on Artificial Intelligence and Statistics*, pages 2021–2031. PMLR, 2020.

[55] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004.

[56] Matthias Seeger. PAC-Bayesian generalisation error bounds for Gaussian process classification. *Journal of machine learning research*, 3(Oct):233–269, 2002.

[57] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. Personalized federated learning using hypernetworks. *arXiv preprint arXiv:2103.04628*, 2021.

[58] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321, 2015.

[59] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. Federated multi-task learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4427–4437, 2017.

[60] Jake Snell and Richard Zemel. Bayesian few-shot classification with one-vs-each Pólya-Gamma augmented Gaussian processes. In *International Conference on Learning Representations*, 2021.

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

[62] Sebastian U Stich. Local SGD converges fast and communicates little. In *International Conference on Learning Representations*, 2018.

[63] Canh T Dinh, Nguyen Tran, and Tuan Dung Nguyen. Personalized federated learning with Moreau envelopes. *Advances in Neural Information Processing Systems*, 33, 2020.

[64] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *arXiv preprint arXiv:2103.00710*, 2021.

[65] Minxue Tang, Xuefei Ning, Yitu Wang, Yu Wang, and Yiran Chen. FedGP: Correlation-based active client selection for heterogeneous federated learning. *arXiv preprint arXiv:2103.13822*, 2021.

[66] Carlos Villacampa-Calvo, Bryan Zaldívar, Eduardo C Garrido-Merchán, and Daniel Hernández-Lobato. Multi-class Gaussian process classification with noisy inputs. *Journal of Machine Learning Research*, 22(36):1–52, 2021.

[67] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. In *International Conference on Learning Representations*, 2019.

[68] Jianyu Wang and Gauri Joshi. Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms. In *ICML Workshop on Coding Theory for Machine Learning*, 2019.

[69] Florian Wenzel, Théo Galy-Fajou, Christian Donner, Marius Kloft, and Manfred Opper. Efficient Gaussian process classification using Pólya-Gamma data augmentation. In *The AAAI Conference on Artificial Intelligence*, pages 5417–5424. AAAI Press, 2019.

[70] Christopher KI Williams and David Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

[71] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

[72] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2594–2602, 2016.

[73] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[74] Feng Yin, Zhidi Lin, Qinglei Kong, Yue Xu, Deshi Li, Sergios Theodoridis, and Shuguang Robert Cui. FedLoc: Federated learning framework for data-driven cooperative localization and location data processing. *IEEE Open Journal of Signal Processing*, 1:187–215, 2020.

[75] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning*, pages 7252–7261. PMLR, 2019.

[76] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. A survey on federated learning. *Knowledge-Based Systems*, 216:106775, 2021.

[77] Michael Zhang, Karan Sapra, Sanja Fidler, Serena Yeung, and Jose M Alvarez. Personalized federated learning with first order model optimization. *arXiv preprint arXiv:2012.08565*, 2020.

[78] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-IID data. *arXiv preprint arXiv:1806.00582*, 2018.

[79] Fan Zhou and Guojing Cong. On the convergence properties of a K-step averaging stochastic gradient descent algorithm for nonconvex optimization. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3219–3227, 2018.

[80] Pan Zhou, Xiaotong Yuan, Huan Xu, Shuicheng Yan, and Jiashi Feng. Efficient meta learning via minibatch proximal update. *Advances in Neural Information Processing Systems*, 32:1534–1544, 2019.

[81] Wennan Zhu, Peter Kairouz, Brendan McMahan, Haicheng Sun, and Wei Li. Federated heavy hitters discovery with differential privacy. In *International Conference on Artificial Intelligence and Statistics*, pages 3837–3847. PMLR, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] .
   (b) Did you describe the limitations of your work? [Yes] . See section 3.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] . See Section 7.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] .
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] . We made basic assumptions relating to Bayesian modeling and PAC. See section 5.
   (b) Did you include complete proofs of all theoretical results? [N/A] .
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] . See section 6 and the Appendix. Code will be provided.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] . See section 6 and the Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] . We report the standard error of the mean in all experiments.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] . In the Appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] . See section 6.
   (b) Did you mention the license of the assets? [No] . We use standard and publically available assets.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] . We provide a new benchmark dataset for PFL with noisy inputs in the supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] . We use standard and publically available assets.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] .
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] .
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] .
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] .