Federated Learning with Uncertainty via Distilled Predictive Distributions

Shrey Bhatt* SHREYB1997@GMAIL.COM
Eightfold AI

Aishwarya Gupta* AISHWARYAG@CSE.IITK.AC.IN
IIT Kanpur

Piyush Rai PIYUSH@CSE.IITK.AC.IN
IIT Kanpur

Editors: Berrin Yanıkoğlu and Wray Buntine

Abstract

Most existing federated learning methods are unable to estimate model/predictive uncertainty since the client models are trained using the standard loss function minimization approach which ignores such uncertainties. In many situations, however, especially in limited data settings, it is beneficial to take into account the uncertainty in the model parameters at each client as it leads to more accurate predictions and also because reliable estimates of uncertainty can be used for tasks, such as out-of-distribution (OOD) detection, and sequential decision-making tasks, such as active learning. We present a framework for federated learning with uncertainty where, in each round, each client infers the posterior distribution over its parameters as well as the posterior predictive distribution (PPD), distills the PPD into a single deep neural network, and sends this network to the server. Unlike some of the recent Bayesian approaches to federated learning, our approach does not require sending the whole posterior distribution of the parameters from each client to the server but only the PPD in the distilled form as a deep neural network. In addition, when making predictions at test time, it does not require computationally expensive Monte-Carlo averaging over the posterior distribution because our approach always maintains the PPD in form a single deep neural network. Moreover, our approach does not make any restrictive assumptions, such as the form of the clients' posterior distributions, or of their PPDs. We evaluate our approach on classification in federated setting, as well as active learning and OOD detection in federated settings, on which our approach outperforms various existing federated learning baselines.

Keywords: Federated learning, Bayesian learning, uncertainty, knowledge distillation

1. Introduction

Federated learning (Kairouz et al., 2021) enables collaborative learning from distributed data located at multiple clients without the need to share the data among the different clients or with a central server. Much progress has been made in recent work on various aspects of this problem setting, such as improved optimization at each client (Li et al., 2020b), improved aggregation of client models at the server (Chen and Chao, 2020), handling

*. Equal contribution. A significant portion of the work was done when Shrey Bhatt was a Masters student at IIT Kanpur.

the heterogeneity in clients' data distributions (Zhu et al., 2021), and also efforts towards personalization of the client models (Mansour et al., 2020).

Most existing formulations of federated learning view it as an optimization problem where the global loss function is optimized over multiple rounds, with each round consisting of point estimation of a loss function defined over the client's local data, followed by an aggregation of client models on a central server. Point estimation, however, is prone to overfitting especially if the amount of training data on clients is very small. Moreover, crucially, such an approach ignores the uncertainty in the client models. Indeed, taking into account the model uncertainty has been shown to be useful not just for improved accuracy and robustness of predictions when the amount of training data is limited, as well as in other tasks, such as out-of-distribution (OOD) detection (Salehi et al., 2021) and active learning (Ahn et al., 2022). In this work, we present a probabilistic approach to federated learning which takes into account the model uncertainty at each client (by learning a *posterior distribution*, i.e., the conditional distribution of the model parameters given the training data), and also demonstrate its effectiveness for other tasks in federated settings where accurate estimates of model uncertainty are crucial, such as OOD detection and active learning in federated setting.

Despite its importance, federated learning in the setting where each client learns a posterior distribution is inherently a challenging problem. Unlike standard federated learning, in this setting, each client needs to estimate the posterior distribution over its weights using Bayesian inference, and also the posterior predictive distribution (PPD) which needed at the prediction stage, which is an intractable problem. Typical ways to address this intractability of Bayesian inference for deep learning models include (1) Approximate Bayesian inference where the posterior distribution of model parameters is usually estimated via approximate inference methods, such as MCMC (Zhang et al., 2019; Izmailov et al., 2021), variational inference (Zhang et al., 2018), or other faster approximations such as modeling the posterior via a Gaussian distribution constructed using the SGD iterates (Maddox et al., 2019), or (2) ensemble methods, such as deep ensembles (Lakshminarayanan et al., 2017) where the model is trained using different initialization to yield an ensemble whose diversity represents the model uncertainty.

The other key challenge for federated learning in this setting is efficiently communicating the client model parameters, which are represented by a probability distribution, to the server, and their aggregation at the server. Note that, unlike standard federated learning, in our setting, to capture client model's uncertainty, each client needs to maintain either a probability distribution over its model weights or an ensemble over the model weights. Both of these approaches make it difficult to efficiently communicate the client models and aggregate them at the server. Some recent attempts towards such settings of federated learning have relied on simplifications such as assuming that the posterior distribution of each client's weights is a Gaussian (Al-Shedivat et al., 2020; Linsner et al., 2021), which makes model communication and aggregation at the server somewhat easier. However, this severely restricts the expressiveness of the client models. In our work, we do not make any assumption on the form of the posterior distribution of the client weights. Another appealing aspects of our federated learning approach is that, at test time, it does not require Monte-Carlo averaging (Bishop, 2006; Korattikara Balan et al., 2015) which is usually required by Bayesian methods (especially for non-conjugate models, such as deep learning models) at

test time, making them slow (essentially, using m Monte-Carlo samples from the posterior makes prediction m times slower). In contrast, our approach leverages ideas from the distillation of posterior predictive distribution (PPD) (Korattikara Balan et al., 2015), using which we are able to represent the entire posterior predictive distribution using a single deep neural network, resulting in fast predictions at test time.

Our contributions are summarized below

- We present a novel and efficient probabilistic framework to federated learning in which each client performs a distillation of its posterior predictive distribution into a single deep neural network. This allows solving the problem of federated learning with client model uncertainty using ideas developed for standard federated learning methods, while still capturing and leveraging model uncertainty.
- Our approach does not make any strict assumptions on the form of the clients' posterior distributions (e.g., Gaussian (Al-Shedivat et al., 2020)) or predictive distributions. Moreover, despite each client learning a posterior distribution, our approach is still fast at test time since it does not require Monte-Carlo averaging (which is akin to averaging over an ensemble) but uses the idea of distribution distillation to represent the posterior predictive distribution (PPD) via a single deep neural network.
- We present various ways to aggregate the clients' predictive distributions at the server, both with as well as without requiring publicly available (unlabeled) data at the server.
- In addition to tasks such as classification and out-of-distribution (OOD) detection, we also show a use case of our approach for the problem of active learning in federated setting (Ahn et al., 2022) where our approach outperforms existing methods.

2. Bayesian Federated Learning via Predictive Distribution Distillation

Unlike standard federated learning where the client model is represented by a single neural network whose weights are estimated by minimizing a loss function using client's data, we consider the setting of federated learning where each client learns a posterior distribution over its weights. The posterior distribution $p(\theta|\mathcal{D})$ is a probability distribution of network weights representing how likely a sample θ explains the training data $\mathcal{D}$. The goal is to efficiently communicate the clients' local posteriors to the server and aggregate these local posteriors to learn a global model that can serve all the clients.

However, since we usually care about predictive tasks, the actual quantity of interest in our probabilistic setting is not the posterior distribution per se, but the posterior predictive distribution (PPD). Given a set of m samples $\theta^{(1)}, \ldots, \theta^{(m)}$ from the posterior, estimated using some training data $\mathcal{D}$, the Monte Carlo approximation of the PPD of of a test input x is defined as $p(y|x, \mathcal{D}) = \frac{1}{m} \sum_{i=1}^{m} p(y|x, \theta^{(i)})$. Note that the PPD can be thought of as an ensemble of m models drawn i.i.d. from the posterior.

Since the PPD is the actual quantity of interest, in our probabilistic federated learning setting, we aim to directly estimate the PPD at each client. However, even estimating and representing the PPD has challenges. In particular, since the PPD is essentially an ensemble of models, storing and communicating such an ensemble from each client to the server can be challenging. To address this issue, we leverage the idea of distribution/ensemble

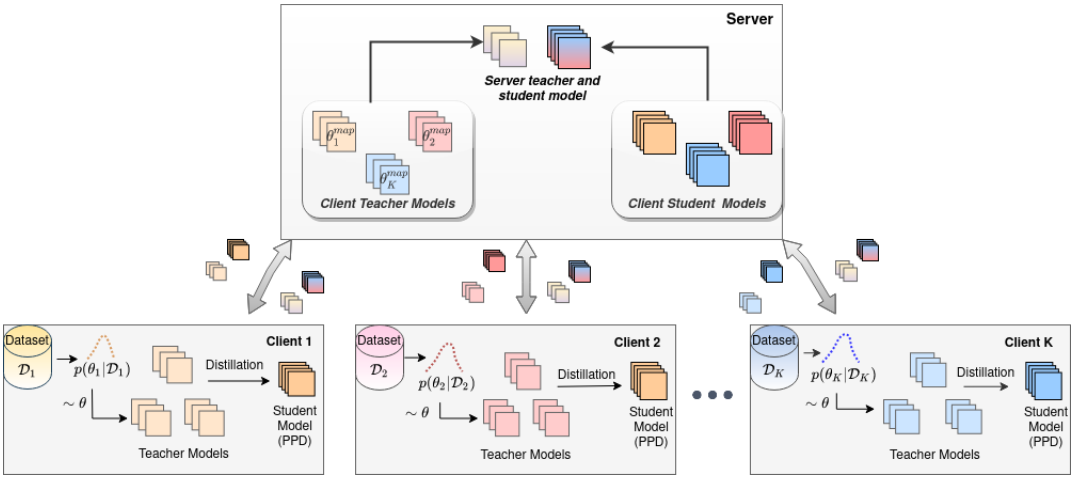

Figure 1: The above figure summarizes our framework. Each client infers the (approximate) posterior distribution by generating the posterior samples (teacher models) which are distilled to give the PPD (student model parameterized by a deep neural network). Each client communicates its MAP teacher sample and the PPD to the server which aggregates them to yield a global teacher sample and the global PPD, both of which are sent back to the clients, which use these quantities for the next round of learning.

distillation (Korattikara Balan et al., 2015), where the PPD of a deep learning model can be efficiently distilled and stored as a single deep neural network. We leverage this distillation idea on each client to represent the client's PPD using a single neural network which can then be communicated and aggregated at the server in pretty much the same way as it is done in standard federated learning. We also note that, although we use the distillation framework proposed in (Korattikara Balan et al., 2015), our approach is general and can leverage various other recently proposed methods for distribution/ensemble distillation.

Our approach can be summarized as follows (and is illustrated in Figure. 1)

1. For each client, we perform approximate Bayesian inference for the posterior distribution of the client model weights using Markov Chain Monte Carlo (MCMC) sampling. This gives us a set of samples from the client's posterior and these samples will be used as teacher models which we distill into a student model. We use stochastic gradient Langevin dynamics (SGLD) sampling (Welling and Teh, 2011) since it gives us an online method to efficiently distill these posterior samples into a student model (step 2 below).

2. For each client, we distill the MCMC samples (teacher models) directly into the posterior predictive distribution (PPD), which is the student model. Notably, in this distillation based approach (Korattikara Balan et al., 2015), the PPD for each client is represented succinctly by a *single* deep neural network, instead of via an ensemble of deep neural network. This makes prediction stage much faster as compared to typical Bayesian approaches.

3. For each client, the teacher model with largest posterior probability (i.e., the MAP sample) from its posterior distribution and the student model representing the client's PPD (both of which are deep neural networks), are sent to the server.

4. The server aggregates the teacher and student models it receives from all the clients. For the aggregation, we consider several approaches described in Sec 2.2.

5. The aggregated teacher and student models are sent back to each client, and the process continues for the next round.

6. We continue steps 1-5 till convergence.

2.1. Posterior Inference and Distillation of Client's PPD

We assume there are K clients with labeled data $\mathcal{D}_1, \ldots, \mathcal{D}_K$, respectively. On each client, we take the Monte Carlo approximation of its posterior predictive distribution (PPD) and distill it into a single deep neural network using an online Bayesian inference algorithm, as done by the Bayesian Dark Knowledge (BDK) approach in (Korattikara Balan et al., 2015). Each iteration of this distillation procedure first generates a sample from the client's posterior distribution using the stochastic Langevin gradient descent (SGLD) algorithm (Welling and Teh, 2011) and incrementally "injects" the sample into a deep neural network $\mathcal{S}$ (referred to as "student") with parameters w, representing a succinct form of the client's (approximate) PPD. This is illustrated by each of the client blocks shown in Figure 1. For client k, assuming the set of samples generated by SGLD to be $\theta_k^{(1)}, \ldots, \theta_k^{(m)}$, this distillation procedure can be seen as learning the parameters w_k of the client k's student model $\mathcal{S}_k$ by minimizing the following loss function (Korattikara Balan et al., 2015) using an unlabeled distillation dataset $\mathcal{D}'_k$ at client k

$$\hat{L}(w_k) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{x' \in \mathcal{D}'_k} \mathbb{E}_{p(y=j|x',\theta_k^{(i)})} \log \mathcal{S}_k(y=j|x', w_k) \tag{1}$$

Note that, in the above equation, $\log \mathcal{S}_k(y=j|x', w_k)$ is the log of the student model output value indicating the predicted probability of the label y taking value j for some input x'. The unlabeled dataset $\mathcal{D}'_k$ can be generated from the original labeled dataset $\mathcal{D}_k$ by adding perturbations to the inputs, as suggested in (Korattikara Balan et al., 2015).

We sketch the full algorithm for optimizing for w_k in the Supplementary Material. We use this algorithm at each client to learn the student model $\mathcal{S}_k$ which represents a compact approximation of the client k's PPD in form of a single deep neural network (as shown in the client block in Figure 1), which can be now communicated to the server just like client models are communicated in standard federated learning algorithms. Note that, as shown in Figure 1, in our federated setting, in addition to weights w_k of its PPD approximation (the student model), each client k also sends the approximate maximum-a-posteriori (MAP) sample θ_k^{MAP} defined as the sample $\theta_k^{(i)}$, $i \in \{1, 2, \ldots, m\}$ with the largest posterior density $p(\theta_k^{(i)}|\mathcal{D}_k)$. Note that θ_k^{MAP} is typically an *approximate* MAP sample since the posterior $p(\theta_k|\mathcal{D})$ itself is approximated using sampling. The overall sketch of our federated learning procedure, which we call FedPPD (Federated Learning via Posterior Predictive Distributions), is shown in Algorithm 1.

Algorithm 1: FedPPD

Data: Number of communication rounds T, total clients K, unlabeled dataset $\mathcal{U} = \{x_i\}_{i=1}^{P}$, server teacher model weights θ_g, server student model weights w_g, client teacher model weights $\{\theta_i\}_{i=1}^{K}$, client student model weights $\{w_i\}_{i=1}^{K}$, number of training samples at client $\{n_i\}_{i=1}^{K}$

Result: Final Server Student Model Weight $w_g^{(T)}$

for *each round* $t = 0, \ldots, T-1$ **do**
  Server broadcasts $\theta_g^{(t)}$ and $w_g^{(t)}$
  for *each client* $i \in \{1, \ldots, K\}$ **do**
    $\theta_i = \theta_g^{(t)}$, $w_i = w_g^{(t)}$
    Update θ_i and w_i locally as per (Korattikara Balan et al., 2015)
  end
  Communicate $\{\theta_i^{MAP}\}_{i=1}^{K}$ and $\{w_i\}_{i=1}^{K}$ to server
  $\theta_g^{(t+1)}$, $w_g^{(t+1)}$ = Server_Update($\{\theta_i^{MAP}\}_{i=1}^{K}, \{w_i\}_{i=1}^{K}, \{n_i\}_{i=1}^{K}$)
end

2.2. Aggregation of Client Models

As described in the previous section, the server receives two models from client k - the (approximate) MAP sample θ_k^{MAP} (the teacher) as well as the (approximate) PPD w_k (the student). We denote the teacher models (approximate MAP samples) from the K clients as $\{\theta_1^{MAP}, \ldots, \theta_K^{MAP}\}$ and the respective student models (approximate PPD) as $\{w_1, \ldots, w_K\}$. These models are aggregated at the server and then sent back to each client for the next round. We denote the server aggregated quantities for the teacher and student models as θ_g and w_g (we use g to refer to "global").

In this work, we consider and experiment with two aggregation schemes on the server. **Simple Aggregation of Client Models:** Our first aggregation scheme (shown in Algorithm 2) computes dataset-size-weighted averages of all the teacher models and all the student models received at the server. Denoting the number of training examples at client k as n_k and $N = \sum_{k=1}^{K} n_k$, we compute $\theta_g = \frac{1}{N}\sum_{k=1}^{K} n_k \theta_k^{MAP}$ and $w_g = \frac{1}{N}\sum_{k=1}^{K} n_k w_k$, similar to how FedAvg algorithm (McMahan et al., 2017) aggregates client models on the server.

Distillation-based Aggregation of Client Models: Our second aggregation scheme goes beyond computing (weighted) averages of models received only from the clients. The motivation behind this approach is that the client models (both teachers as well as students) received at the server may not be diverse enough to capture the diversity and heterogeneity of the clients (Chen and Chao, 2020). To address this issue, this approach (shown in Algorithm 3) first fits two probability distributions, one over the K teacher models and the other over the K student models received from the clients. It then uses these distributions to generate M *additional* client-*like* teacher models and student models. Using the actual teacher models (resp. student models) and the additionally *generated* teacher models (resp. student models), we perform knowledge distillation on the server to compute the global

Algorithm 2: Server_Update (Average)

Data: client teacher model's MAP estimate $\{\theta_i^{MAP}\}_{i=1}^K$, client student model weights $\{w_i\}_{i=1}^K$, number of training samples at client $\{n_i\}_{i=1}^K$

Result: Resultant Teacher Model $\overline{\theta}$, Student Model $\overline{w}$

$N = \sum_{i=1}^K n_i$ `/* total number of samples */`

$\overline{\theta} = \frac{1}{N} \sum_{i=1}^K n_i \theta_i^{MAP}$

$\overline{w} = \frac{1}{N} \sum_{i=1}^K n_i w_i$

teacher model θ_g and the global student model w_g. This server-side distillation procedure requires an *unlabeled* dataset $\mathcal{U}$ on the server. Applying the actual and generated teacher models (resp. student models) on the unlabeled dataset $\mathcal{U}$ gives us pseudo-labeled data $\mathcal{T}$ where each pseudo-label is defined as the averaged prediction (softmax probability vector) obtained by applying the actual and generated teacher models (resp. student models) to an unlabeled input. For the distillation step, we finally run the Stochastic Weighted Averaging (SWA) algorithm (Izmailov et al., 2018) using the pseudo-labeled data $\mathcal{T}$ and the simple aggregation of the client models as initialization. Both θ_g and w_g can be obtained by following this procedure in an identical manner. Recently, this idea was also used in Federated Bayesian Ensemble (FedBE) (Chen and Chao, 2020) to learn the global model.

The two aggregation schemes for server-side updates are shown in Algorithm 2, and 3. Note that, among the two aggregation schemes, only Algorithm 3 assumes the availability of unlabeled dataset at the server. Also, owing to high computation capacity, server can compute θ_g and w_g in parallel for all the aggregation schemes; incurring no additional delays in communication rounds.

3. Related Work

Federated learning has received considerable research interest recently. The area is vast and we refer the reader to excellent surveys (Li et al., 2020a; Kairouz et al., 2021) on the topic for a more detailed overview. In this section, we discuss the works that are the most relevant to our work.

While standard federated learning approaches assume that each client does point estimation of its model weights by optimizing a loss function over its own data, recent work has considered posing federated learning as a posterior inference problem where a global posterior distribution is inferred by aggregating local posteriors computed at each client. FedPA (Al-Shedivat et al., 2020) is one such recent approach which performs approximate inference for the posterior distribution of each client's weights. However, it assumes a restrictive form for the posterior (Gaussian), as also assumed in some other recent works (Liu et al., 2021; Guo et al., 2023). Moreover, the method needs to estimate the covariance matrix of the Gaussian posterior, which is difficult in general and approximations are needed. Moreover, although FedPA estimates the (approximate) posterior on each client, due to efficiency/communication concerns, at the server, it only computes a point estimate (the

Algorithm 3: Server_Update (Distill)

Data: Unlabeled dataset $\mathcal{U}$, client teacher model's MAP estimate $\{\theta_i^{MAP}\}_{i=1}^K$, client student model weights $\{w_i\}_{i=1}^K$, number of training samples at client $\{n_i\}_{i=1}^K$

Result: Resultant Teacher Model θ_g, Student Model w_g

$N = \sum_{i=1}^K n_i$ `/* total number of samples */`

$\overline{\theta} = \frac{1}{N}\sum_{i=1}^K n_i \theta_i^{MAP}, \quad \overline{w} = \frac{1}{N}\sum_{i=1}^K n_i w_i$

begin

Construct global teacher model distribution $p(\theta|D)$ from $\{\theta_i^{MAP}\}_{i=1}^K$ `/* using Gaussian approximate */`

Sample M additional teachers and form teacher ensemble $E_T = \{\theta_j \sim p(\theta|\mathcal{D})\}_{j=1}^M \cup \{\overline{\theta}\} \cup \{\theta_i\}_{i=1}^K$

Annotate $\mathcal{U}$ using E_T to generate pseudo-labeled dataset $\mathcal{T}$

Distill E_T knowledge to $\overline{\theta}$ using SWA : $\theta_g = SWA(\overline{\theta}, E_T, \mathcal{T})$

end

Similarly follow the above steps with $\{w_i\}_{i=1}^K$ and $\overline{w}$ to get w_g

mean) of the global posterior. Thus, even though the approach is motivated from a Bayesian setting, in the end, it does not provide a posterior distribution or a PPD for the global model.

Recently, (Linsner et al., 2021) presented methods for uncertainty quantification in federated learning using a variety of posterior approximation methods for deep neural networks, such as Monte Carlo dropout (Gal and Ghahramani, 2016), stochastic weight averaging Gaussian (SWAG) (Maddox et al., 2019), and deep ensembles (Lakshminarayanan et al., 2017). These approaches, however, also suffer from poor quality of approximation of the posterior at each client. (Lee et al., 2020) also propose a Bayesian approach for federated learning. However, their approach also makes restrictive assumptions, such as the distribution of the gradients at each of the clients being jointly Gaussian.

Probabilistic/Bayesian approaches for federated learning have also been proposed in recent work in the context of learning *personalized* models at each client, using ideas such as Gaussian Process Achituve et al. (2021) and variation inference Zhang et al. (2022). In contrast to these works, our setting is aimed at learning a global model that can be served to all the clients.

Instead of a simple aggregation of client models at the server, FedBE (Chen and Chao, 2020) uses the client models to construct a distribution at the server and further distills this distribution into a single model using distillation. However, FedBE only performs point estimation ignoring any uncertainty in the client models. Another probabilistic approach to federated learning Thorgeirsson and Gauterin (2020) fits a Gaussian distribution using the client models, and sends the mean of this Gaussian to each client for the next round of client model training. This approach too does not estimate a posterior at each client, and thus ignores the uncertainty in client models.

In the context of Bayesian learning, recent work has also explored federated versions of Markov Chain Monte Carlo sampling algorithms, such as stochastic gradient Langevin dynamics sampling (Lee et al., 2020; El Mekkaoui et al., 2021). While interesting in their own right in terms of performing MCMC sampling in federated settings, these methods are not designed with the goal of real-world applications of federated learning, where fast prediction and compact model sizes are essential.

Among other probabilistic approaches to federated learning, recent work has explored the use of latent variables in federated learning. In Louizos et al. (2021), a hierarchical prior is used on client model's weights where the prior's mean is set to the server's global model, and additional latent variables can also be used to impose other structures, such as sparsity of client model weights. However, these approaches do not model the uncertainty in the client model.

Some of the recent work on federated learning using knowledge distillation is also relevant. Note that our work leverages the ideas of teacher-student distillation, both at the clients (when learning a representation of the PPD using a single deep neural network), as well as in our second aggregation strategy where server-side distillation is used for learning the global model. In federated learning, the idea of distillation has been used in other works as well, such as federated learning when the client models are of different sizes and the (weighted) averaging does not make sense due to the different size/architecture of different client models (Zhu et al., 2021). Moreover, server-side distillation of client models Lin et al. (2020) has in general been found to outperform simpler aggregation schemes, such as FedAvg.

Recently, Kassab and Simeone (2022) proposed an approach for Bayesian federated learning which represents each client's posterior using a set of particles. However, in each round, all the particles needs to be sent to the server, making both communication as well as server-side aggregation very expensive.

4. Experiments

In this section, we compare our client uncertainty-driven probabilistic federated learning approach with various relevant baselines on several benchmark datasets. We report results on the following tasks: (1) Classification in federated setting, (2) Active Learning in federated setting, and (3) OOD detection on each client. In this section, we refer to our approach with simple averaging on the server side as FedPPD, and the variant with distillation based aggregation on the server side as FedPPD+Distill. We have also made our code publicly available at https://github.com/aishgupta/fedppd.

4.1. Experimental Setup

4.1.1. Baselines

We compare our methods with the following baselines

(1) **FedAvg** (McMahan et al., 2017) is the standard federated algorithm in which the local models of the participating clients are aggregated at server to compute a global model which is then sent back to all the clients for initialization in the next round.

(2) **FedBE** (Chen and Chao, 2020) is another state-of-the-art baseline which provides a more robust aggregation scheme. Instead of only averaging the client models at the server, a probability distribution is fit using the client models, several other models are generated from this probability distribution, and then the client models as well as the generated models are distilled into a single model to yield the global model at the server, which is sent to all the clients for initialization in the next round. Note however that the clients in FedBE only perform point estimation of their weights unlike our approach which estimates the posterior distribution and the PPD of each client.

(3) **Federated SWAG** (Linsner et al., 2021) is a Bayesian federated learning algorithm which is essentially based on a federated extension of the SWAG Maddox et al. (2019) which is an efficient Bayesian inference algorithm for deep neural networks. However, Federated SWAG relies on a simplification that it executes standard federated averaging for all except the last round and in the last round, the SWAG algorithm is invoked at each client to yield a posterior. Also note that Federated SWAG requires Monte-Carlo sampling at test time (thus relying on ensemble based slow prediction) unlike our method which only requires a single neural network to make the prediction.

In addition to the above baselines, in the supplementary material, we also provide a comparison with **FedPA** (Al-Shedivat et al., 2020), a Bayesian federated learning method, which estimates local posteriors (assumed to be Gaussian) over the client weights, and aggregates them at the server to form an approximate global posterior.

4.1.2. Datasets

We evaluate and compare our approach with baseline methods on four datasets: MNIST (LeCun and Cortes, 2010), FEMNIST (Cohen et al., 2017), and CIFAR-10/100 (Krizhevsky et al., 2009). MNIST comprises of images of handwritten digits categorized into 10 classes. It has a total of 60,000 images for training and 10,000 images for testing. FEMNIST consists of images of handwritten characters (digits, lowercase, and uppercase alphabets resulting in total of 62 classes) written by multiple users. It has a total of 80,523 images written by 3,550 users. CIFAR-10 consists of 32×32 dimensional RGB images categorised into 10 different classes. It has a total of 50,000 images for training and 10,000 images for testing. CIFAR-100 is similar to CIFAR-10 but has 100 distinct classes.

4.1.3. Model Architecture and Configurations

In all our experiments, the student model has a larger capacity compared to teacher model as it is modeling the PPD by distilling multiple models drawn from the posterior distribution. We have used a customized CNN architecture for both teacher and student model on MNIST, FEMNIST and CIFAR-10 dataset, with student model being deeper and/or wider than its corresponding teacher model. For CIFAR-100, ResNet-18 and ResNet-34 are used as the teacher and student model, respectively.

In all our experiments, we consider $K = 10$ clients with data heterogeneity (experimental results for IID setting are reported in the Supplementary Material). Each client holds a small non-i.i.d. subset of training data - approximately 2000 samples for FEMNIST, CIFAR-10 and CIFAR-100 and around 500 samples for MNIST. Except for the FEMNIST data where we have used the Leaf (Caldas et al., 2018) benchmark to distribute data among clients

(excluding digits to increase class imbalance), for all other datasets clients strictly maintains a small subset of all the classes (at most 2 major classes for MNIST and CIFAR-10 and at most 20 major classes for CIFAR-100) For a fair comparison, we run our method and all the baselines for 200 rounds on all the datasets (except MNIST where we run it for 100 rounds) and train local client model for 10 epochs in each round. Also, we assume complete client participation i.e. all the clients are considered in each round. However, we tune the learning rate, momentum and weight decay for each method independently. For FedBE and FedPPD+Distill, we run additional 20 and 50 epochs at the server for distillation on CIFAR/MNIST and FEMNIST datasets, respectively.

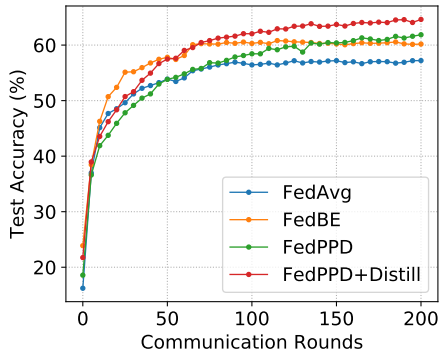

Figure 2: Convergence of all the methods on CIFAR-10 dataset

4.2. Tasks

Classification We evaluate FedPPD (its two variants) and the baselines on several classification datasets and report the accuracy on the respective test datasets. The results are shown in Table 1. We also show the convergence of all the methods on CIFAR-10 in Figure 2 (similar plots for other datasets are provided in Supplementary Material). Both the variants of FedPPD outperform the other baselines on all the datasets. As compared to the best performing baseline, our approach yields an improvement of 4.44% and 7.08% in accuracy on CIFAR-10 and CIFAR-100, respectively. On MNIST and FEMNIST datasets too, we observe noticeable improvements. The improvements across the board indicate that FedPPD and its variants are able to leverage model uncertainty to yield improved predictions especially when the amount of training data per client is small, which is the case with the experimental settings (as mentioned in Sec 4.1.3). We also observe that in cases where there is significant heterogeneity in the data distribution across the different clients (on CIFAR-10 and CIFAR-100), the performance gains offered by FedPPD and its variants are much higher as compared to the baselines. On other simpler tasks like MNIST dataset and FEMNIST with the data distribution being roughly similar across different clients, FedPPD results in reasonable gains in the performance. We also quantify the calibration error of our approach and all the baselines models and report the results in Table 2 where FedPPD and its variant have the least error on the test dataset.

The improved performance of our algorithm FedPPD can be attributed to the following reasons: (1) ability to incorporate model uncertainty which helps us compute the predictive uncertainty, whereas other approaches rely on point estimates of the model parameters; (2)

although FedSWAG also captures model uncertainty, it uses a crude Gaussian approximation of the posterior whereas FedPPD do not make any such strict assumptions.

Model	MNIST	FEMNIST	CIFAR-10	CIFAR-100
FedAvg	97.74	87.40	57.20	47.02
FedAvg+SWAG	97.75	87.45	57.34	47.07
FedBE	97.82	88.12	60.18	47.52
FedPPD	97.85	**88.81**	61.86	53.00
FedPPD+Distill	**98.08**	88.80	**64.62**	**54.60**

Table 1: Federated classification test accuracies on benchmark datasets

	CIFAR-10			CIFAR-100		
Model	ECE	MCE	BS	ECE	MCE	BS
FedAvg	16.88	24.71	0.60	28.83	42.26	0.78
FedAvg+SWAG	16.69	23.16	0.60	29.07	44.36	0.78
FedBE	19.26	27.54	0.59	31.89	45.92	0.80
FedPPD	**4.31**	**6.62**	0.50	13.92	**21.19**	0.63
FedPPD+Distill	10.83	16.71	**0.49**	**13.57**	23.69	**0.61**

Table 2: Expected Calibration Error (ECE), Maximum Calibration Error (MCE) and Brier-score (BS) of all the models on CIFAR-10 and CIFAR-100

Federated Active Learning In active learning, the goal of the learner is to iteratively request the labels of the most informative inputs instances, add these labeled instances to the training pool, retrain the model, and repeat the process until the labeling budget remains. Following (Ahn et al., 2022), we extend our method and the baselines to active learning in federated setting using entropy of the predictive distribution of an input x as the acquisition function. The entropy-based acquisition function for input x is computed as $I(x) = -\sum_{i=1}^{C} p(y = i|x) \log p(y = i|x)$ (C refers to the number of classes) and is used as a score function. In federated active learning setting (we provide a detailed sketch of the federated active learning algorithm in the Supplementary Material), each client privately maintains a small amount of labeled data and a large pool of unlabeled examples. In each round of active learning, clients participate in federated learning with their currently labeled pool of data until the global model has converged. Now, each client uses the global model to identify a fixed number (budget) of the most informative inputs among its pool of unlabeled input based on the highest predictive entropies $I(x)$, which are then annotated (locally maintaining data privacy) and added to the pool of labeled examples. Now, with this, next round of active learning begins, where clients will participate in federated learning and use the global model to expand their labeled pool. This process continues until either the unlabeled dataset has been exhausted completely or desired accuracy has been achieved. For a fair comparison, we have run federated active learning on CIFAR-10 dataset with same parameters for all the approaches. We start active learning with 400 labeled and 3200 unlabeled samples at each client and use a budget of 400 samples in every round of active learning. For federated learning, we use the same hyperparameters as for the classification experiments. We stop federated active learning once all the clients have exhausted their

unlabeled dataset and show the results in Figure 3 where FedPPD and its variant attain the best accuracies among all the methods.

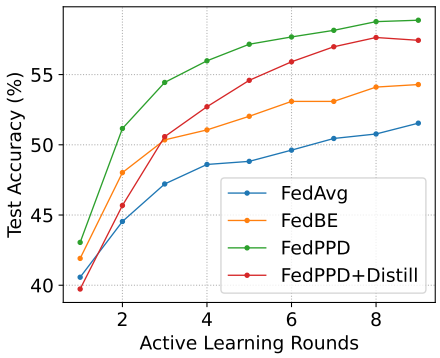

Figure 3: Federated Active Learning on CIFAR-10. Note: FedAvg+SWAG performed almost similarly to FedAvg on this task as well so we skip it from the plot.

Out-of-distribution (OOD) detection We also evaluate FedPPD and its variant, and the other baselines, in terms of their ability to distinguish between Out-of-Distribution (OOD) data and data used during training phase (in-distribution data). For this, given any sample x to be classified among k distinct classes and model weights θ (or PPD for our approach), we compute Shannon entropy of the model's predictive distribution for the input x and compute the AUROC (Area Under ROC curve) metric. We use KMNIST as OOD data for models trained on FEMNIST, and SVHN for CIFAR-10/CIFAR-100 models. Note that, to avoid class imbalance, we sample an equal amount of data for both the distributions (out and in) and repeat it 5 times. We report the results in Table 3 where FedPPD and its variant consistently result in better AUROC scores on all the datasets validating its robustness and accurate estimates of model uncertainty. In addition to OOD detection, we also apply all the methods for the task of identifying the correct predictions and incorrect predictions based on the predictive entropies. For this task too, FedPPD and its variant outperform the other baselines as shown in Table 4.

Model	FEMNIST	CIFAR-10	CIFAR-100
FedAvg	0.957 ± 0.003	0.728 ± 0.013	0.703 ± 0.011
FedAvg+SWAG	0.956 ± 0.003	0.728 ± 0.013	0.704 ± 0.011
FedBE	0.966 ± 0.003	0.728 ± 0.006	0.669 ± 0.009
FedPPD	$\mathbf{0.983 \pm 0.003}$	0.701 ± 0.007	0.698 ± 0.009
FedPPD+Distill	0.949 ± 0.003	$\mathbf{0.765 \pm 0.006}$	$\mathbf{0.784 \pm 0.008}$

Table 3: AUROC score for OOD/in-domain data detection

5. Conclusion and Discussion

Leveraging model uncertainty in federated learning has several benefits as we demonstrate in this work. To achieve this, we developed a uncertainty-driven approach to federated

Model	FEMNIST	CIFAR-10	CIFAR-100
FedAvg	0.846 ± 0.011	0.742 ± 0.011	0.792 ± 0.003
FedAvg+SWAG	0.845 ± 0.009	0.743 ± 0.010	0.800 ± 0.004
FedBE	$\mathbf{0.863 \pm 0.005}$	0.753 ± 0.007	0.789 ± 0.005
FedPPD	0.862 ± 0.008	0.755 ± 0.007	0.814 ± 0.003
FedPPD+Distill	0.853 ± 0.013	$\mathbf{0.769 \pm 0.006}$	$\mathbf{0.823 \pm 0.002}$

Table 4: AUROC score for correct/incorrect data detection

learning by leveraging the idea of distilling the posterior predictive into a single deep neural network. In this work, we consider a specific scheme to distill the PPD at each client. However, other methods that can distill the posterior distribution into a single neural network (Wang et al., 2018; Vadera et al., 2020) are also worth leveraging for probabilistic federated learning. Another interesting future work will be to extend our approach to settings where different clients could possibly be having different model architectures. Finally, our approach first generates MCMC samples (using SGLD) and then uses these samples to obtain the PPD in form of a single deep neural network. Recent work has shown that it is possible to distill an ensemble into a single model without explicitly generating samples from the distribution (Ratzlaff and Fuxin, 2019). Using these ideas for uncertainty-driven probabilistic federated learning would also be an interesting future work.

Acknowledgments

AG acknowledges the Prime Minister's Research Fellowship (PMRF) for the support. PR acknowledges support from Google Research India.

References

Idan Achituve, Aviv Shamsian, Aviv Navon, Gal Chechik, and Ethan Fetaya. Personalized federated learning with gaussian processes. *Advances in Neural Information Processing Systems*, 34:8392–8406, 2021.

Jin-Hyun Ahn, Kyungsang Kim, Jeongwan Koh, and Quanzheng Li. Federated active learning (f-al): an efficient annotation strategy for federated learning. *arXiv preprint arXiv:2202.00195*, 2022.

Maruan Al-Shedivat, Jennifer Gillenwater, Eric Xing, and Afshin Rostamizadeh. Federated learning via posterior averaging: A new perspective and practical algorithms. In *International Conference on Learning Representations*, 2020.

Christopher M Bishop. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

Hong-You Chen and Wei-Lun Chao. Fedbe: Making bayesian model ensemble applicable to federated learning. In *International Conference on Learning Representations*, 2020.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

Khaoula El Mekkaoui, Diego Mesquita, Paul Blomstedt, and Samuel Kaski. Federated stochastic gradient langevin dynamics. In *Uncertainty in Artificial Intelligence*, pages 1703–1712. PMLR, 2021.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

Han Guo, Philip Greengard, Hongyi Wang, Andrew Gelman, Yoon Kim, and Eric Xing. Federated learning as variational inference: A scalable expectation propagation approach. In *The Eleventh International Conference on Learning Representations*, 2023.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 876–885, 2018.

Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Gordon Wilson. What are bayesian neural network posteriors really like? In *International Conference on Machine Learning*, pages 4629–4640. PMLR, 2021.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

Rahif Kassab and Osvaldo Simeone. Federated generalized bayesian learning via distributed stein variational gradient descent. *IEEE Transactions on Signal Processing*, 70:2180–2192, 2022.

Anoop Korattikara Balan, Vivek Rathod, Kevin P Murphy, and Max Welling. Bayesian dark knowledge. *Advances in Neural Information Processing Systems*, 28, 2015.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Seunghoon Lee, Chanho Park, Song-Nam Hong, Yonina C Eldar, and Namyoon Lee. Bayesian federated learning over wireless networks. *arXiv preprint arXiv:2012.15486*, 2020.

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, 2020a.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020b.

Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363, 2020.

Florian Linsner, Linara Adilova, Sina Däubener, Michael Kamp, and Asja Fischer. Approaches to uncertainty quantification in federated deep learning. In *ECML PKDD Workshop on Parallel, Distributed, and Federated Learning*, pages 128–145. Springer, 2021.

Liangxi Liu, Feng Zheng, Hong Chen, Guo-Jun Qi, Heng Huang, and Ling Shao. A bayesian federated learning framework with online laplace approximation. *arXiv preprint arXiv:2102.01936*, 2021.

Christos Louizos, Matthias Reisser, Joseph Soriaga, and Max Welling. An expectation-maximization perspective on federated learning. *arXiv preprint arXiv:2111.10192*, 2021.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

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.

Neale Ratzlaff and Li Fuxin. Hypergan: A generative model for diverse, performant neural networks. In *International Conference on Machine Learning*, pages 5361–5369. PMLR, 2019.

Mohammadreza Salehi, Hossein Mirzaei, Dan Hendrycks, Yixuan Li, Mohammad Hossein Rohban, and Mohammad Sabokrou. A unified survey on anomaly, novelty, open-set, and out-of-distribution detection: Solutions and future challenges. *arXiv preprint arXiv:2110.14051*, 2021.

Adam Thor Thorgeirsson and Frank Gauterin. Probabilistic predictions with federated learning. *Entropy*, 23(1):41, 2020.

Meet Vadera, Brian Jalaian, and Benjamin Marlin. Generalized bayesian posterior expectation distillation for deep neural networks. In *Conference on Uncertainty in Artificial Intelligence*, pages 719–728. PMLR, 2020.

Kuan-Chieh Wang, Paul Vicol, James Lucas, Li Gu, Roger Grosse, and Richard Zemel. Adversarial distillation of bayesian neural network posteriors. In *International conference on machine learning*, pages 5190–5199. PMLR, 2018.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.

Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.

Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical stochastic gradient mcmc for bayesian deep learning. In *International Conference on Learning Representations*, 2019.

Xu Zhang, Yinchuan Li, Wenpeng Li, Kaiyang Guo, and Yunfeng Shao. Personalized federated learning via variational bayesian inference. In *International Conference on Machine Learning*, pages 26293–26310. PMLR, 2022.

Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *International Conference on Machine Learning*, pages 12878–12889, 2021.