Supplementary: Federated Learning with Uncertainty via Distilled Predictive Distributions

1. Learning Posterior Predictive Distribution (PPD) locally at each client

At each client, we aim to distill the Monte Carlo approximation of its PPD into a single neural network. We do this using an online approach, similar to (Korattikara Balan et al., 2015). We maintain two deep neural networks on each client: 1) the first neural network is optimized using SGLD on client's private dataset to draw samples from its posterior (these samples denote the collection of teacher models on each client), and 2) the second neural network represents the distilled version of the client's Monte Carlo approximation of its PPD (student model). In each local iteration at client, we draw a sample (i.e., a teacher model) from its posterior distribution and distill it into the student model which represents the PPD. This incremental distillation process, when all the teacher models are distilled into the student model, ultimately gives the student model representing the PPD (in form of a single deep neural network) at the client. We summarize the client updates in Algorithm 1.

Algorithm 1: Local update at client k

```
Data: Dataset $\mathcal{D}_k = (x_i, y_i)_{i=1}^{N_k}$, batch-size $M_k$, number of local iterations $V$, teacher
      model weights $\theta$, student model weights $w$, teacher learning rate at iteration $v$,
      $\alpha^v$, student learning rate at iteration $v$, $\beta^v$, teacher prior hyperparameters $\gamma_k$,
      student prior hyperparameters $\mu_k$
for $v$ = 0 ... $V-1$ do
    /* teacher model update */
    Sample a minibatch $B$ of size $M_k$
    Sample Gaussian Noise $z_v \sim \mathcal{N}(0, \alpha^v I)$
    $\theta_k^{v+1} = \theta_k^v + \frac{\alpha^v}{2}\nabla_{\theta_k}(\log p(\theta_k^v|\gamma_k) + \frac{N_k}{M_k}\sum_{(x,y)\in\mathcal{B}} \log p(y|x, \theta_k^v)) + z_v$
    Generate minibatch $B'$ of size $M_k$
    /* student model update */
    Add Gaussian Noise to $B'$
    $w_k^{v+1} = w_k^v + \beta^v \nabla_{w_k}(\frac{1}{M_k}\sum_{x\in B'}\sum_c p(y=c|x, \theta_k)\log p(y=c|x, w_k^v) + \log p(w_k^v|\mu_k))$.
end
```

2. Federated Active Learning

Active learning is an iterative process which aids a learner in achieving desired performance with limited number of labeled input instances. In each iteration of active learning, the learner identifies the most informative inputs, requests their labels, adds them to its pool of labeled instances and retrain the model with augmented labeled dataset. This process repeats until the labeling budget is not fully exhausted. Similarly, in federated active learning (Ahn et al., 2022), active learning can be performed on each client using the global model (informative of global data distribution) instead of its local model to identify the most helpful instances. Thus, in each iteration of federated active learning, each client identifies the most helpful inputs, annotates it locally (or using oracle preserving data privacy), adds it to its local dataset and participates in federated learning until the convergence of the global model. In our work, we use entropy of the model's output as the score function to identify the most helpful inputs, though other predictive uncertainty based score functions used in active learning can be employed as well. Also, note that to compute $p(y|x)$, FedPPD uses posterior predictive distribution whereas the other baseline methods like FedAvg uses the point-estimate of the global model. Our federated active learning algorithm is sketched in detail in Algorithm 2.

Algorithm 2: Federated Active Learning

```
Data: Number of active rounds $A$, client id $\{1, 2, 3, \ldots, k\}$, client labeled dataset
      $\{\mathcal{L}_i\}_{i=1}^{k}$, client unlabeled dataset $\{\mathcal{U}_i\}_{i=1}^{k}$, budget per round $\mathcal{B}$, set of inputs to
      be annotated $S'$, global server model $\theta$
for each round $r = 0, \ldots, A-1$ do
    for each client $i = 0, \ldots, k$ do
        $S_i = \{\}$
        for each input $x \in \mathcal{U}$ do
            $p(y|x) = \theta_r(x)$
            $I(x) = -\sum_{i=1}^{C} p(y = y_i|x) \log p(y = y_i|x)$
            $S_i = S_i \cup \{(x, I(x))\}$
        end
        Select subset $S_i'$ of size $\mathcal{B}$ from $S_i$ with maximum entropy and get it annotated
        $\mathcal{L}_i = \mathcal{L}_i \cup S_i'$    $\mathcal{U}_i = \mathcal{U}_i - S_i'$
    end
    $\theta_{r+1}$ = Updated global model using federated learning on $\{\mathcal{L}_i\}_{i=1}^{k}$
end
```

3. Experimental Setup

We now provide the implementation details of the experiments mentioned in main paper. The code for our method is available in the form of a zip file as a part of the supplementary material. The instructions for executing the experiments are also provided in the file readme.md.

Model architecture We evaluate all the variants of FedPPD and baseline algorithms on MNIST, FEMNIST, and CIFAR-10 using customized CNNs and use ResNets for CIFAR-100. To have a fair comparison, the architecture of the teacher model in FedPPD and client model in all the baseline approaches is the same. We provide the architecture details of the teacher and student model for all datasets in Table 1 and 2, respectively.

MNIST	FEMNIST	CIFAR-10
Conv2D$(5 \times 5, 10)$	Conv2D$(5 \times 5, 32)$	Conv2D$(5 \times 5, 6)$
MaxPool(2×2)	MaxPool(2×2)	MaxPool(2×2)
Conv2D$(5 \times 5, 20)$	Conv2D$(5 \times 5, 64)$	Conv2D$(5 \times 5, 16)$
Dropout2D(0.5)	Dropout2D(0.5)	MaxPool(2×2)
MaxPool(2×2)	MaxPool(2×2)	Linear(120)
Linear(50)	Linear(128)	Linear(84)
Dropout(0.5)	Dropout(0.5)	Linear(10)
Linear(10)	Linear(52)	

Table 1: Model Architecture for Teacher Network and Baselines Methods

MNIST	FEMNIST	CIFAR-10
Conv2D$(5 \times 5, 20)$	Conv2D$(5 \times 5, 50)$	Conv2D$(5 \times 5, 16)$
MaxPool(2×2)	MaxPool(2×2)	MaxPool(2×2)
Conv2D$(5 \times 5, 40)$	Conv2D$(5 \times 5, 100)$	Conv2D$(5 \times 5, 16)$
Dropout2D(0.5)	Dropout2D(0.5)	MaxPool(2×2)
MaxPool(2×2)	MaxPool(2×2)	Linear(256)
Linear(100)	Conv2D$(5 \times 5, 200)$	Linear(128)
Dropout(0.5)	Dropout2D(0.5)	Linear(10)
Linear(10)	MaxPool(2×2)	
	Linear(1600)	
	Dropout(0.5)	
	Linear(180)	
	Dropout(0.5)	
	Linear(52)	

Table 2: Model Architecture for Student Networks

Hyperparameters We tune the hyperparameters (learning rate and weight decay) on each dataset for all the variants of FedPPD and baseline methods. The optimal learning rate of the teacher and student model during local learning in FedPPD are $\{0.045, 0.055\}$, $\{0.050, 0.085\}$ and $\{0.055, 0.020\}$ on MNIST, FEMNIST and CIFAR respectively. In case of FedPPD with distillation at server, the teacher and student learning rates during local training are $\{0.045, 0.055\}$, $\{0.060, 0.085\}$ and $\{0.055, 0.020\}$ on MNIST, FEMNIST and CIFAR respectively. Also, during distillation at server, teacher and student model are updated using SWA optimizer with learning rates of $\{0.0010, 0.0010\}$ and $\{0.0015, 0.0025\}$ for MNIST/FEMNIST and CIFAR-10/100 respectively.

Baseline We compare our approach with FedAvg (McMahan et al., 2017) and FedBE (Chen and Chao, 2020) on MNIST, FEMNIST, CIFAR-10 and CIFAR-100 and the results are presented in the main paper. Here, we provide the model convergence plot for all the approaches on FEMNIST and CIFAR-100 in Fig 1, showing the superior performance of FedPPD and its variants as compared to the other baselines.

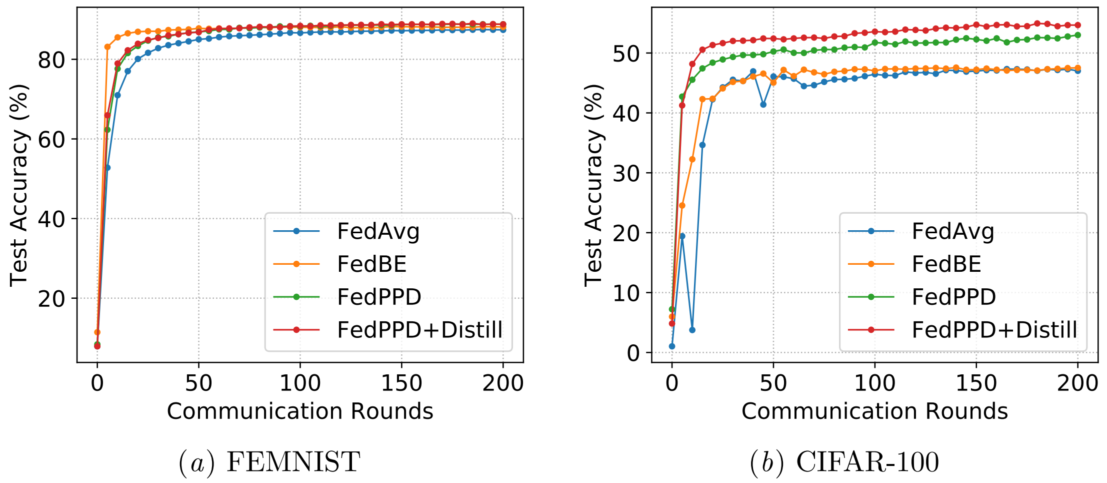

Figure 1: Convergence of different federated approaches

We also compare our method against FedPA (Al-Shedivat et al., 2020) (approximates the posterior distribution by a Gaussian) using their publicly available code obtained from (`https://github.com/google-research/federated/tree/master/posterior_averaging`). We extend their implementation to our experimental setting as described in main paper. The results of FedPA in our experimental setup are either comparable or worse than FedAvg. We report the results of FedPA on all the datasets in Table 3. A possible reason for the poor performance of FedPA could be because the Gaussian approximation of global posterior may not be accurate enough, and moreover since, at the server, FedPA only computes the Gaussian posterior's mode/mean but not the covariance, using only the mode/mean ignores the uncertainty, leading to poorer predictions.

	MNIST	FEMNIST	CIFAR-10	CIFAR-100
FedPA	96.86	86.47	50.88	42.18

Table 3: Performance of FedPA on test dataset

IID Data Distribution We now consider a setting in which the data is distributed among clients in IID fashion. We have experimented on CIFAR-10 and CIFAR-100 dataset and compared our proposed approach with the baseline methods. We have assigned a subset of 4000 randomly selected labeled images to 10 client and have maintained a small subset of 10000 unlabeled images as the proxy dataset on the server for FedBE and FedPPD+Distill. We have also considered complete client participation i.e. all the 10 clients participate in every communication round of the federated learning. Also, for all the remaining hyperparameters, we have used same values as we did for the non-IID setting. The results of the experiments are reported in Table 4 where FedPPD and its variant clearly outperforms the baseline methods. This shows that FedPPD and FedPPD+Distill result in improved performance even when the data distribution among clients is homogeneous.

	FedAvg	FedBE	FedPPD	FedPPD+Distill
CIFAR-10	73.80%	73.99%	74.91%	**75.26%**
CIFAR-100	53.38%	53.16%	**63.67%**	63.43%

Table 4: Classification performance of the global model on the test dataset for IID data distribution among clients

Resources used We ran all our experiments on Nvidia 1080 Ti GPUs with 12 GB of memory. We have implemented our method in PyTorch and utilized its multiprocessing library to spawn multiple threads for parallel computation.

4. Potential Limitations/Future Work

We have shown the efficacy and robustness of our approach against multiple baseline methods on various datasets. Even though, to obtain (an approximation to) the PPD at each client, we use a specific approach based on stochastic gradient MCMC (SGMCMC) combined with knowledge distillation, our work provides a general framework for Bayesian federated learning where we can use a variety of methods at each client to obtain the posterior/PPD approximation, and then leverage techniques developed for standard federated learning. The key is to represent the PPD approximation via a single deep neural network. In our work, we use the vanilla SGMCMC at the clients which sometimes can have convergence issues. However, recent process on SGMCMC has led to more robust variants of SGMCMC which can be employed under our framework for better performance.

Generation of posterior samples using MCMC/SGMCMC and then distilling them into a student model (even using an online procedure like us) can be slow, especially since MCMC methods can sometimes exhibit slow convergence. One possible avenue of future work could be to represent the posterior of the model implicitly and distill it without having to explicitly generate samples from it (Ratzlaff and Fuxin, 2019).

References

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.

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

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

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.