Supplement for "A Bayesian Approach for Personalized Federated Learning in Heterogeneous Settings"

In this supplementary material, we first include pseudo-code of the algorithm used for training our framework. Then, we provide definitions and results used in the privacy analysis of our method. We show model calibration metrics and present results demonstrating our method is well-calibrated. We also discuss the details about the alignment dataset, AD, its affect on the performance and the communication and computation cost of the procedure.

A ALGORITHM

The pseudo-code of the algorithm used in the FedBNN method is included in the Algorithm 1. The Algorithm 1 works in the setting when there is a server connected to N clients with each client i having local dataset $\mathcal{X}_i$ of size n_i drawn from the local data distribution $\boldsymbol{D}_i$, and the server has an auxilliary unlabelled dataset called AD. The output of the algorithm is the set of personalized models Φ_i parameterized by $\mathcal{W}_i$ for each client i. All $\mathcal{W}_i$'s, instead of being point estimates, are determined by a posterior distribution $\mathbb{P}(\mathcal{W}_i|.)$ which is learnt from the data via variational inference. As mentioned in the Section 3.2, the learning procedure first optimizes the prior parameters by minimizing Equation 4 and then learns the posterior parameters keeping the prior fixed by minimizing Equation 5.

Algorithm 1 FedBNN Algorithm

Input: number of clients N, number of global communication rounds T, number of local epochs E, weight vector $[w_1, w_2, \dots w_N]$, noise parameter γ
Output: Personalised BNNs $\{\Phi_i | i \in [1, N]\}$, parameterized by $\mathcal{W}_i \sim \mathbb{P}(\mathcal{W}_i|\mathcal{X})$
Server Side -
X = AD
for $t = 1$ **to** T **do**
  Select a subset of clients $\mathcal{N}_t$
  for each selected client $i \in \mathcal{N}_t$ **do**
    $\Phi_i(\mathbf{X}) =$ **LocalTraining**$(t, \bar{\Phi}(\mathbf{X})^{(t-1)}, \mathbf{X})$
  end for
  $\bar{\Phi}(\mathbf{X})^{(t)} = \sum_{j=1}^{N} w_j \Phi_j(\mathbf{X})$
end for
Return $\Phi_1(T), \Phi_2(T) \dots \Phi_N(T)$
LocalTraining$(t, \bar{\Phi}(\mathbf{X})^{(t-1)}, \mathbf{X})$
Run inference on **X** to obtain $\Phi_i(\mathbf{X})$
Generate $\Phi_i^{\text{corrected}}(\mathbf{X}) = \gamma \bar{\Phi}(\mathbf{X})^{(t-1)} + (1-\gamma)\Phi_i(\mathbf{X})$
for each prior epoch **do**
  Minimize CrossEntropy$(\Phi_i^{\text{corrected}}(\mathbf{X}), \bar{\Phi}(\mathbf{X})^{(t-1)})$ to obtain prior parameters ψ of the BNN Φ_i
end for
for each local epoch **do**
  Minimize $\text{KL}[q(\mathcal{W}_i|\theta)||p(\mathcal{W}_i;\psi^*)] - \mathbb{E}_{q(\mathcal{W}_i|\theta)}[log\mathbb{P}(\mathcal{X}_i|\mathcal{W}_i)]$ over $\{\theta : q(\mathcal{W}_i|\theta) \in \mathcal{Q}\}$ to obtain θ^*
end for
$\mathbb{P}(\mathcal{W}_i|\mathcal{X}) \approx q(\mathcal{W}_i|\theta^*)$
Obtain K Monte-carlo samples $\mathcal{W}_i^{(j)} : j \in [1, K]$ from $\mathbb{P}(\mathcal{W}_i|\mathcal{X})$
Compute $\Phi_i(\mathbf{X}) = \frac{1}{K}\sum_{j=1}^{K} \Phi_i(\mathbf{X}; \mathcal{W}_i^{(j)})$
Return $\Phi_i(\mathbf{X})$

B PRIVACY ANALYSIS

Some known results on differential privacy that are used to determine the privacy loss of our algorithm are given in this section.

A generalization of differential privacy is concentrated differential privacy(CDP). And an alternative form of concentrated differential privacy called zero-concentrated differential privacy(zCDP) was proposed to enable tighter privacy analysis (Bun & Steinke, 2016). We will also use the zCDP notion of privacy for our analysis. The relationship between standard DP and zCDP is shown below.

Proposition B.1 ((ϵ, δ)-DP and ρ-zCDP)**.** *For a randomized algorithm $\mathcal{M}$ to satisfy (ϵ, δ)-DP, it is sufficient for it to satisfy $\frac{\epsilon^2}{4log\frac{1}{\delta}}$-zCDP. And a randomized algorithm $\mathcal{M}$ that satisfies ρ-zCDP, also satisfies (ϵ', δ)-DP where $\epsilon' = \rho + \sqrt{4\rho log\frac{1}{\delta}}$.*

As opposed to the notion of DP, the zCDP definition provides tighter bounds for the total privacy loss under compositions, allowing better choice of the noise parameters. The privacy loss under the serial composition and parallel composition incurred under the definition of zCDP was proved by (Yu et al., 2019) and is recalled below.

Proposition B.2 (Sequential Composition)**.** *Consider two randomized mechanisms, $\mathcal{M}_1$ and $\mathcal{M}_2$, if $\mathcal{M}_1$ is ρ_1-zCDP and $\mathcal{M}_2$ is ρ_2-zCDP, then their sequesntial composition given by $(\mathcal{M}_1(), \mathcal{M}_2())$ is $(\rho_1 + \rho_2)$-zCDP.*

Proposition B.3 (Parallel Composition)**.** *Let a mechanism $\mathcal{M}$ consists of a sequence of k adaptive mechanisms, $(\mathcal{M}_1, \mathcal{M}_2, \dots \mathcal{M}_k)$ working on a randomized partition of the $D = (D_1, D_2, \dots D_k)$, such that each mechanism $\mathcal{M}_i$ is ρ_i-zCDP and $\mathcal{M}_t : \prod_{j=1}^{t-1} \mathcal{O}_j \times D_t \to O_t$, then $\mathcal{M}(D) = (\mathcal{M}_1(D_1), \mathcal{M}_2(D_1), \dots \mathcal{M}_k(D_k))$ is $\max_i \rho_i$-zCDP.*

After computing the total privacy loss by an algorithm using the tools described above, we can determine the variance of the noise parameter σ for a set privacy budget. The relationship of the noise variance to privacy has been shown in prior works by (Dwork & Roth, 2014; Yu et al., 2019) and is given below.

Definition B.4 (L_2 Sensitivity)**.** For any two neighboring datasets, D and D' that differ in at most one data point, L_2 sensitivity of a mechanism $\mathcal{M}$ is given by maximum change in the L_2 norm of the output of $\mathcal{M}$ on these two neighboring datasets

$$\Delta_2(\mathcal{M}) = \sup_{D, D'} ||\mathcal{M}(D) - \mathcal{M}(D')||_2.$$

Proposition B.5 (Gaussian Mechanism)**.** *Consider a mechanism $\mathcal{M}$ with L_2 sensitivity Δ, if on a query q, the output of $\mathcal{M}$ is given as $\mathcal{M}(x) = q(x) + \mathcal{N}(0, \sigma^2)$, then $\mathcal{M}$ is $\frac{\Delta^2}{2\sigma^2}$-zCDP.*

C CALIBRATION

Model calibration is a way to determine how well the model's predicted probability estimates the model's true likelihood for that prediction. Well-calibrated models are much more important when the model decision is used in critical applications like health, legal etc. because in those cases managing risks and taking calculated actions require a confidence guarantee as well. Visual tools such as reliability diagrams are often used to determine if a model is calibrated or not. In a reliability diagram, model's accuracy on the samples is plotted against the confidence. A perfectly calibrated model results in an identity relationship. Other numerical metrics that could be used to measure model calibration include Expected Calibration Error (ECE) and Maximum Calibration Error (MCE). ECE measures the expected difference between model confidence and model accuracy whereas MCE measures the maximum deviation between the accuracy and the confidence. The definitions and empirical formulas used for calculating ECE and MCE are as given below.

$$\text{ECE} = \mathbb{E}_{\hat{P}}[\mathbb{P}(\hat{Y} = Y|\hat{P} = p) - p],$$

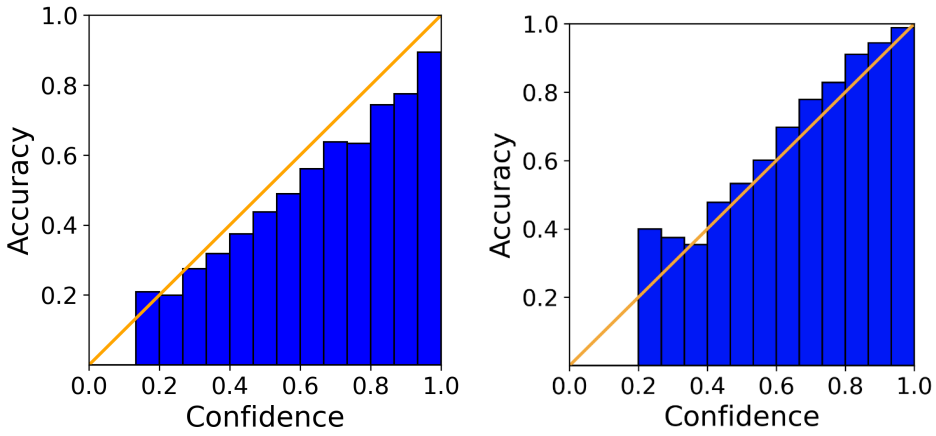
$$\text{MCE} = \max_{p \in [0,1]} |\mathbb{P}(\hat{Y} = Y|\hat{P} = p) - p|.$$

Empirically,

$$\text{ECE} = \sum_{i=1}^{M} \frac{|B_i|}{n} |\text{accuracy}(B_i) - \text{confidence}(B_i)|,$$

$$\text{MCE} = \max_{i \in [1,M]} |\text{accuracy}(B_i) - \text{confidence}(B_i)|,$$

where B_i is a bin with set of indices whose prediction confidence according to the model falls into the range $\left(\frac{i-1}{M}, \frac{i}{M}\right)$. Figure 2 shows the reliability diagram along with the ECE and MCE scores for our method measured on MNIST and CIFAR-10 dataset in the non-IID data setting.



(a) **Dataset**: CIFAR-10, **ECE**: 0.070, **MCE**: 0.134 (b) **Dataset**: MNIST, **ECE**: 0.032, **MCE**: 0.156

Figure 2: Reliability diagrams and scores showing model calibration. Figure (a) is for the results corresponding to the CIFAR-10 dataset and Figure (b) for MNIST dataset.

D ALIGNMENT DATASET (AD)

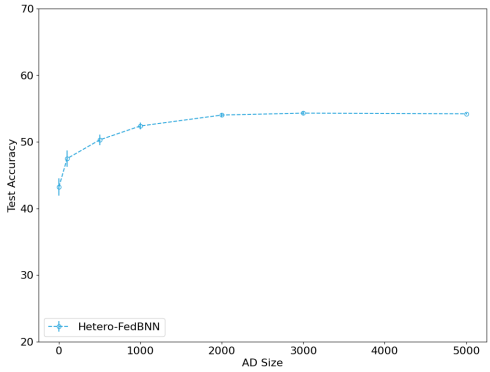


Figure 3: Ablation study comparing the affect of AD size on the performance. The included results are for CIFAR-10 dataset in the small data setting with non-IID partitions and heterogeneous clients.

In FedBNN, the alignment dataset (AD) is used to achieve collaboration across clients. Since the only assumption on AD is for it to be of the same domain as the target application, there is no practical constraint on obtaining the AD in real-world settings. In many cases it could be obtained from web, for example images from common datasets in Huggingface, texts from Wikipedia, Reddit etc. The use of AD is not different from how several other methods use an additional dataset for augmentation. The effect of size of AD on the performance of models is demonstrated in Figure 3 for CIFAR-10 dataset in the small data and non-IID setting. In that figure, we observe that when the size of AD is small the performance of the model is low but as the size of AD increases the performance increases up to a point and becomes constant afterwards. The number of data points in AD that are required to achieve good improvement in the model performance is small and practical.

We also vary the distribution of the AD being used and test the final performance of the models and report it in Table 2. We run these experiments on 20 clients for CIFAR-10 dataset where each client

Table 2: Effect of varying distribution of AD on the clients' performance for the non-IID seeting with CIFAR-10 dataset and 20 clients where each client has data for the 5 different classes.

Architecture Setting	Local Training	CIFAR10(10)	CIFAR10(8)	CIFAR10(5)	CIFAR10(2)	SVHN
Homogeneous Architectures	64.3± 0.36	72.7 ± 0.15	69.7 ± 0.28	68.8 ± 0.97	67.2 ± 1.5	70.1 ± 0.18
Heterogeneous Architectures	61.2± 0.17	71.6 ± 0.93	68.4 ± 0.80	68.8 ± 1.4	68.1 ± 1.9	69.3 ± 0.8

had access to only 5 of the 10 classes and each client belonged to the medium data setting. For the first experiment, we use a held-out dataset from the CIFAR-10 data as AD but vary the composition of the dataset by changing the distribution of the classes present in the AD, for example, CIFAR10(10) is composed of all 10 classes present in the CIFAR-10 dataset but CIFAR10(2) is composed of only 2 out of the 10 classes present in the AD and likewise. We also test the performance of our method when a significantly different dataset SVHN consisting of the colored house number images is used. Table 2 suggests that the performance of the method even with different datasets as AD *always* improves and that the gain between local training and the proposed procedure is better highlighted in the heterogeneous architecture settings, since there local client capacities and model architectures differ significantly and clients are able to utilize the peer knowledge to learn better models locally. We observed that even for different and dissimilar data distributions in AD, it is possible to obtain a value for the parameter γ such that the final performance of the local client model with collaboration is better than the model independently trained locally on the client. The parameter γ controls the amount of global knowledge to be incorporated on each client and helps in regularization of the client models which enables them to generalize better and perform better on the test data.

E Communication and Computation Efficiency

Communication Cost In FedBNN, each global communication round requires that the server sends the alignment dataset to all the clients and the clients upload the outputs of their respective models on the common dataset AD. Since AD is a publicly available dataset, AD could be transmitted to the clients by specifying the source and the indices, and does not really needs to be communicated across the channel. The client output on AD, on the other hand, depends on the number of instances in AD, let's call it K, therefore, the total communication cost in each round of our method is $O(K)$. As shown in Figure 3, having $K = 2000$ gives a good performance. The communication cost between the clients and the server, thus, is also invariant of the number of model parameters which tends to run in millions. This allows our method to be much more communication efficient as compared to the conventional FL algorithms that transmit model parameters in each communication round, making it practically more useful.

Computation Cost Similarly, the computation cost of a FL procedure involves the costs incurred in local training at the individual clients and the cost of aggregation at the server, both of which are discussed below.

- **Server-side computation cost** The server side computation cost arises from the need to aggregate knowledge obtained from individual clients. In the state-of-the-art bayesian FL algorithms, the server aggregates posterior distributions for each weight parameter in the neural network obtained from various clients. The number of such weight parameters typically run in millions. In our method we do not aggregate the parameter distributions but achieve collaboration by aggregating the client outputs on the AD (with size 2000), thus the server side computation cost in our method is many orders of magnitude lower than the conventional methods and does not depend on the number of model parameters. This makes our method much more efficient and scalable than existing federated bayesian solutions.
- **Client-side computation cost** The client-side computation cost is mostly determined by the cost of training a Bayesian Neural Network at the client side, which in turn depends on the type of inference procedure used for obtaining the posterior distribution over weights for each parameter of the neural network. In the proposed work, the method used for inference is Bayes by Backprop which uses the gradient computations similar to backpropagation(BP) algorithm to obtain the posterior distributions where the posterior distributions

Table 3: Performance comparison as a function of the privacy guarantee.

Privacy (ϵ) per round	Test Accuracy
≈ 1	75.5 %
≈ 0.1	71.3 %
≈ 0.01	68.6 %
≈ 0.001	62.2 %
≈ 0.0001	59.6 %

Table 4: Test accuracy comparison with more number of clients (500) in the setting.

Method	Test Accuracy
pFedGP	53.2 ± 0.4
pFedBayes	52.9 ± 0.8
Ours(Homo)	56.1 ± 0.3
Ours(Hetero)	54.7 ± 1.0

are characterized by the mean and standard deviation. A re-parameterization trick is used to compute the mean and std of the distributions from the backpropagated gradients. Thus the cost for obtaining the posterior distributions is similar to the cost of backpropagation. Moreover, since the method only uses gradient updates, the optimizations used for SGD like asynchronous SGD etc. could be readily used for obtaining the posteriors. An unrelated but similar algorithm in (Hernández-Lobato & Adams, 2015) does probabilistic backpropagation to train BNNs and shows that the average run time of probabilistic BP is not higher than that of BP.

To summarize, the communication cost and the server-side computation cost of the proposed method is orders of magnitude lower than that of the other Bayesian baseline methods. On the other hand, the client-side computation cost is determined by the inference procedure used to obtain the posterior distributions and for which Bayes by Backprop provides an efficient mechanism. Several works in the recent past have discussed the use of related Bayesian inference based methods for training uncertainty-aware transformers (Zheng et al., 2023; Tian et al., 2023; Maged & Xie, 2022) proving that Bayesian methods are not limited to use in simpler models. And therefore, our framework can also be extended to apply in settings where much larger neural networks are required.

F Additional Experiments

Privacy vs Performance Since the amount of noise required to be added to the client's outputs via the Gaussian Mechanism is directly proportional to the guaranteed privacy, we test the affect of the privacy guarantee on the performance of the proposed framework by comparing the performance of the method with varying ϵ and $\delta = 10^{-4}$. The results are reported in Table 3. We observe that, as expected, when we reduce the amount of privacy loss in each iteration by adding more noise to the clients' outputs going to the server, the performance of the method drops. However the drop in performance in all the cases is not drastic as the clients can tune the level of personalization or global knowledge required by appropriately setting the parameter γ in Equation 3.

More clients To test the performance of the proposed method when a large number of clients are involved in the setup, we did additional experiments with 500 clients and non-IID setting with 5 classes per client in the medium data setting on the CIFAR-10 dataset where in each communication round only 10% of the clients are selected for participation and $\gamma = 0.7$. The obtained results at the end of 200^{th} communication round are reported in Table 4. We observe that the homogeneous version of our method is better than the baselines by a significant margin and the heterogeneous version is slightly better than the baselines.