Neural Networks 185 (2025) 107135

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Full Length Article

Federated learning meets Bayesian neural network: Robust and uncertainty-aware distributed variational inference ☆

Pengfei Li [a,b], Qinghua Hu [a,*], Xiaofei Wang [a]

[a] College of Intelligence and Computing, Tianjin University, Tianjin, 300350, China
[b] School of Intelligence Science and Engineering, Qinghai Minzu University, Xining, 810007, China

ARTICLE INFO

Dataset link: https://github.com/lpf111222/FedUAB/

Keywords:
Federated learning
Bayesian neural network
Variational inference
Uncertainty

ABSTRACT

Federated Learning (FL) is a popular framework for data privacy protection in distributed machine learning. However, current FL faces some several problems and challenges, including the limited amount of client data and data heterogeneity. These lead to models trained on clients prone to drifting and overfitting, such that we just obtain suboptimal performance of the aggregated model. To tackle the aforementioned problems, we introduce a novel approach explicitly integrating Bayesian neural networks (BNNs) into the FL framework. The proposed approach is able to enhance the robustness. We refer to this approach as FedUAB, standing for FL with uncertainty-aware BNNs. In the FedUAB algorithm, each FL client independently trains a BNN using the Bayes by backprop algorithm. The weights of approximating model are modeled as Gaussian distributions, which mitigates the overfitting issue and also ensures better data privacy. Besides, we apply novel methods to overcome other key challenges in the fusion of BNNs and FL, such as selecting an optimal prior distribution, aggregating weights characterized by Gaussian forms across multiple clients, and rigorously managing weights variances. In the simulation of a FL environment, FedUAB demonstrated superior performance with both its server-side global model and client-side personalized models, outperforming traditional FL and other Bayesian FL methods. Moreover, it possesses the capability to quantify and decompose uncertainties. We have open-sourced our project at https://github.com/lpf111222/FedUAB/.

1. Introduction

1.1. Background

As data volumes surge and concerns over data privacy intensify, the centralized execution of sophisticated learning tasks by a single server has become impractical. Federated Learning (FL) has been introduced as a solution to address these challenges, with continued advancements in subsequent years. The body of work in question can be categorized as a specialized form of client-based training, emphasizing the utilization of local data without the need for server-side transmission, thereby ensuring robust user privacy protection. By enabling on-device training and limiting uploads to computational outcomes, the risk of sensitive information exposure is mitigated (Wahab et al., 2021; Yang et al., 2019).

In contrast to centralized machine learning, the diversity in user preferences, geographic distribution, and data collection methodologies among FL clients inherently generates the non-identically distributed (NonIID) attribute of client-side training data. Hence, there are still many important challenges in the realm of FL, in this research we focus on the following three issues:

(1) **Client drift**: The divergence of client-specific models from the overarching global model is attributed to the non-identical and non-independent distribution of data across clients (Du et al., 2024; Jing et al., 2023). Such conditions can result in diminished performance and reduced accuracy for the global model, as the local updates may not effectively contribute to the global learning objective (Zhao et al., 2018).
(2) **Client overfitting**: Constraints on the volume of data on individual clients can result in overfitting during clients model training. Under overfitting, training loss decreases rapidly, rendering it ineffective to obtain effective updates of model weights (Li & Zhan, 2021). The overfitting model of FL is typically characterized by over-confidence and lack of robustness, exhibiting poor performance on out-of-distribution (OoD) samples.

☆ This work is partly supported by National Natural Science Foundation of China under Grant 61925602, U23B2049 and 62266035.
* Corresponding author.
E-mail addresses: lipengfei@qhmu.edu.cn (P. Li), huqinghua@tju.edu.cn (Q. Hu), xiaofeiwang@tju.edu.cn (X. Wang).

https://doi.org/10.1016/j.neunet.2025.107135
Received 28 June 2024; Received in revised form 21 November 2024; Accepted 6 January 2025
Available online 13 January 2025

(3) **Data privacy protection**: FL typically employs the addition of random noise as a method to protect privacy, such as in the case of differential privacy (DP) (Wei et al., 2020), but applying random interference to the weights of the model necessarily results in an unpredictable decline in performance (Wang et al., 2023).

This paper addresses the aforementioned challenges from the perspective of BNNs. By leveraging BNNs, the robustness of the model is enhanced, and the comprehensive set of methods proposed in this study ensures the seamless application of BNNs in FL, contributing modestly to the advancement of FL algorithms.

1.2. Solutions and challenges

This study proposes a comprehensive set of solutions that integrate and apply both FL and BNNs. BNNs distinguish it from Deep Neural Networks (DNNs) through the probabilistic assignment of weights, in contrast to the fixed-value approach characteristic of DNNs. This integration of probabilistic modeling with neural networks results in probabilistic inference outputs, allowing for the assessment of uncertainty in the predictions. The probabilistic inference approach of BNNs also significantly enhances its robustness when dealing with limited data, noise, and OoD (Goan & Fookes, 2020).

Capitalizing on the benefits of BNNs, we explicitly integrate BNNs into FL, as opposed to simulating BNNs effects through methods such as dropout. This approach aims to address client drift and overfitting challenges resulting from NonIID. By doing so, we thereby potentially enhance client data privacy protection through the use of probabilistic model weights. Additionally, this model maintains the uncertainty quantification capability of BNNs. We have termed this solution the **FedUAB** (FL with Uncertainty-Aware BNNs).

However, training BNNs within the architecture of FL faces numerous problems, and indeed, research in this area is relatively rare. BNN is not currently research hotspot due to issues such as limited performance gains. Adapting it to be suitable for distributed FL scenarios involves addressing the following challenges:

(1) How to design an effective loss function for training BNNs with limited training data on FL clients?
(2) How to aggregate, on the FL server, model weights in probabilistic distribution form received from different clients?
(3) What is the optimal prior distribution?
(4) How can we ensure the effectiveness of FedUAB, while also considering high accuracy and the ability to quantify uncertainty?

This paper will address the aforementioned challenges one by one, and demonstrate the efficacy of FedUAB. The Fig. 1 below provides an overall of FedUAB, the architecture is similar to the traditional FL, with the distinctions lying in the training of the BNNs, weights aggregation, and assessing uncertainty, which are precisely the innovative aspects of this study.

1.3. Contribution

The FedUAB proposed in this study integrates BNNs within the FL framework, offering the following advantages and contributions:

- We propose a method for **training BNNs on FL clients**. Each client maintains its own BNN, with each weight drawn from a Gaussian distribution. Clients train BNNs model employing the Bayes by Backprop algorithm, which updates weights by optimizing against a compression metric known as variational free energy. The stochastic nature of probabilistic sampling in this variational inference approach also incidentally safeguards the privacy of the training data.
- A novel FL **weights aggregation** method is developed based on the probability conflation theory. The weights of multiple clients (every weight following a Gaussian distribution) aggregate to the new global weights that remain Gaussian distribution. These aggregated new weights serve as the initial weights and **prior** for the subsequent FL training round.
- The trained FedUAB model can obtain output with **higher accuracy and quantized uncertainty** through Monte Carlo sampling, which can even decompose the uncertainty into aleatoric uncertainty and epistemic uncertainty.
- The experiments have substantiated the aforementioned advantages of FedUAB and provided practical examples of quantifying and decomposing uncertainty.

The rest of the paper is structured as follows: we review and summarize the relevant works on BNNs and its integration with FL in Section 2. Section 3 combines Fig. 1 to outline the overview of FedUAB. Section 4 describes the technical details of how BNNs are implemented in FL. The experimental results are shown in Section 5. Finally, Section 6 summarized the whole work.

2. Related work

This section provides a concise overview of BNNs and their quantification of uncertainty, followed by a synthesis of the research advancements in the integration of BNNs with FL. Special emphasis is placed on studies closely related to our work, thereby highlighting the significance of our research contributions.

2.1. BNNs

BNNs are stochastic neural networks trained using a Bayesian approach, it is performed by giving the network either a stochastic weights or stochastic activation to simulate multiple possible models (Goan & Fookes, 2020). Several studies have articulated the robustness and data efficiency of BNNs across various applications, especially when the training data are scarce.

Bayesian inference furnishes an approach to parameter estimation and the quantification of uncertainty within deep learning frameworks (Jospin et al., 2022). Bayesian inference algorithms adapted for deep learning primarily include Markov Chain Monte Carlo (MCMC) and variational inference.

The model parameters estimation in MCMC algorithms is achieved by generating a sequence of samples from the target distribution, typically the posterior, which are derived from the Markov chain constructed through Monte Carlo sampling (Hastings, 1970). It necessitates extensive sampling and the retention of these samples post-training, which proves to be computationally costly for the majority of deep learning models (Chandra & Simmons, 2024).

Variational inference is a method for learning an approximation of the posterior, it has become widely adopted and demonstrates superior scalability compared to MCMC algorithms. Instead of enabling direct sampling from the precise posterior, employs an approximate distribution Q_θ, known as the variational distribution, which is controlled by a set of parameters θ. The parameters θ are optimized to minimize the discrepancy between the variational distribution Q_θ and the true posterior (Blei et al., 2017), the loss function in this optimization learning process is typically the variational free energy (Graves, 2011). The algorithms that adapt the aforementioned variational inference to deep learning are primarily probabilistic backpropagation (Hernández-Lobato & Adams, 2015) and Bayes by Backprop (Blundell et al., 2015), with the latter being more commonly employed.

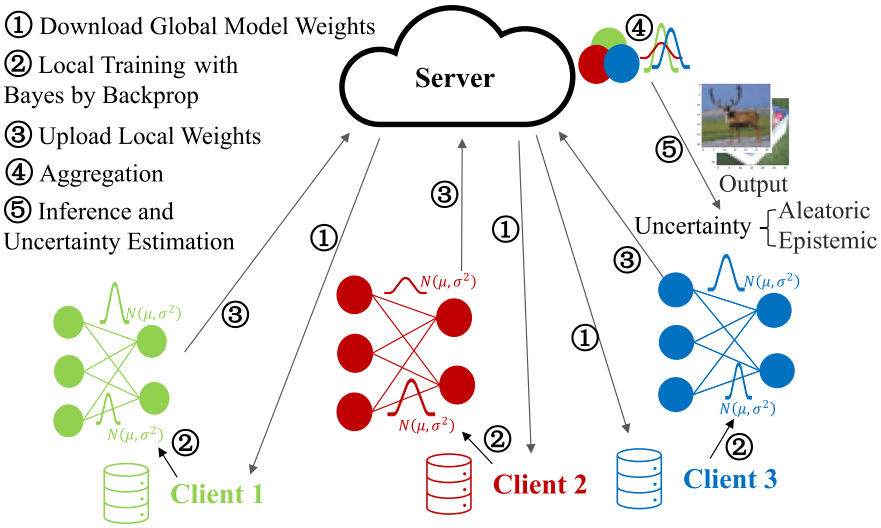

Fig. 1. The schematic diagram of FedUAB. FedUAB iteratively trains the model through the repeated cycles of steps ① - ④, while ⑤ represents the inference output and uncertainty decomposition on the samples using the trained model.

2.2. Uncertainty evaluation by BNNs

BNNs facilitate the quantification of uncertainty in model outputs, with numerous studies providing detailed expositions on the process (Blundell et al., 2015; Shridhar et al., 2018), and uncertainty is distinctly categorized into aleatoric and epistemic components (Depeweg et al., 2018; Kwon et al., 2018).

Owing to the high computational complexity of BNNs, researchers have explored approximations of BNNs to obtain uncertainty evaluation at a reduced cost, with the most widely recognized method being the utilization of dropout as an approximation to Bayesian variational inference (Gal & Ghahramani, 2016), this approach is even applicable within edge computing scenarios that have limited computational resources (Qendro et al., 2021). Additionally, methods such as Approximately Bayesian Ensembling (Pearce et al., 2020) and Laplace Approximation (Lindinger et al., 2022) are utilized.

2.3. Bayesian FL

The integration of BNNs with FL is an emerging research field. More broadly, the incorporation of Bayesian learning principles into the study of FL is categorized under Bayesian Federated Learning (BFL). Within BFL methodologies, Bayesian learning techniques leverage prior knowledge about data to train FL model with a limited number of samples from clients, and also enable the quantification of uncertainties. Furthermore, it augments FL's capabilities in addressing privacy, communication, and heterogeneity (Cao et al., 2023). In the following paragraph, we will provide a synthesis of seminal studies within this domain.

On the client side, implementations of BNNs can be achieved through the utilization of MCMC methods, as exemplified by FedPPD (Bhatt et al., 2024) and FedPop (Kotelevskii et al., 2022), or variational inference, as seen in pFedBayes (Zhang et al., 2022) and BPFed (Chen et al., 2023), during client training. Some approaches, pFedGP (Achituve et al., 2021) and PFNM (Yurochkin et al., 2019) also employ Bayesian methods on the client side for non-parametric models.

When implementing BNNs on the server side in FL, a representative work involves model ensemble during parameter aggregation, such as FedBE (Chen & Chao, 2020). Another approach, exemplified by FedPA (Al-Shedivat et al., 2020), involves Bayesian posterior decomposition, which dissects the global model's posterior into a combination of local models.

The practical significance of BFL is that it offers a feasible exploration pathway for FL in aspects such as personalization, heterogenization, and hierarchical structuring, with personalization being the most extensively realized. Personalization is a highlight of many BFL endeavors, many papers introduced in the previous paragraph have 'personalized' in their titles, due to the inherent nature of Bayesian learning to combine prior information with training data to derive posterior distributions, it provides a wealth of scenarios for personalized research within FL. For example, utilizing the server's global model as a prior for training client-side personalized models(such as pFedBayes), or setting up client-specific, localized priors that reflect individual characteristics, such as those employed in pFedBreD (Shi et al., 2024).

2.4. Positioning FedUAB in the BFL

On the foundation of reviewing the related work from BNNs to BFL, this paper introduces FedUAB, which pertains to the direct training of BNNs on the client side within FL. FedUAB algorithm employs variational inference-based Bayes by Backprop. Its relationship with existing research and its positioning within the broader BFL technological framework are depicted in Fig. 2.

It is evident that FedUAB is indeed closely related to pFedBayes, as both utilize variational inference, specifically the Bayes by Backprop algorithm suitable for deep learning, for training client-local BNNs on the client, and both approaches incorporate the server's global model parameters as priors during client training. The innovative aspects of FedUAB, as previously discussed in our introduction under contributions, involve the server model weight aggregation based on probability conflation theory, and the uncertainty quantification capabilities of the BNNs.

Furthermore, unlike pFedBayes and BPFed that emphasize personalization, FedUAB strikes a balance between the performance of the server-side global model and the client-side personalized models, as demonstrated in the subsequent experimental results.

2.5. Complexity comparison: Time, space, and communication

Considering the higher complexity of BFL compared to traditional FL, this section compares the time, space, and communication complexity of the aforementioned BFL methods.

We compare the computational time, space, and communication load between our proposed FedUAB and some BFL implementation

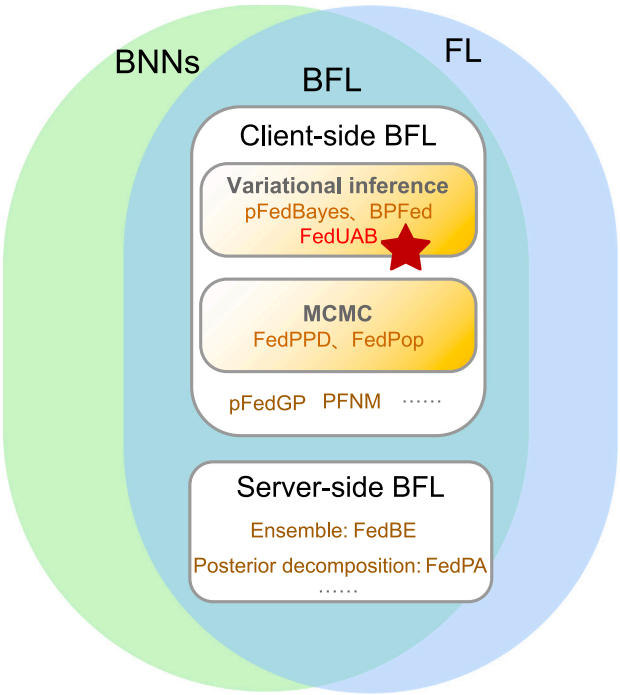

Fig. 2. A technical classification schematic of BFL-related work, with FedUAB proposed in this paper denoted by a red pentagram.

Table 1
Complexity comparison.

FL types	Method	Complexity		
		Time	Space	Communication
Traditional FL	FedAvg	1	1	1
BFL by MCMC	FedPop	×M	×M	1
BFL by variational inference	pFedBayes FedUAB	×2Z	×2Z	×2

methods with traditional FL. The data in Table 1 presents the comparison by showing the multiple of the baseline expenditure required by other BFL methods over the traditional FL.

In the Table 1, M denotes the number of samples taken during the MCMC process. For BFL implementations based on MCMC, such as FedPop, the primary computational and storage expenses are incurred by the M sampling instances and the preservation of the sampled data, in the communication phase, it is necessary to transmit the model weights that have been processed by the client.

BFL implementations utilizing variational inference, such as our FedUAB and the closely related pFedBayes, require both mean and variance for each model weight, effectively doubling the number of parameters. Consequently, the communication load is at least twice that of traditional FL. During the forward pass of Monte Carlo sampling, FedUAB and similar methods employ local reparameterization trick, performing computations for both mean and variance, which at least doubles the computational and storage requirements. With Z iterations of Monte Carlo sampling in training and inference, the time and space complexity are typically scaled by a factor of $2Z$.

Typically, the number of iterations Z in variational inference is significantly smaller than the number of samples M in MCMC(in the experiments with FedPop, $M = 50$). For instance, in our FedUAB, an experiment with $Z = 5$ has achieved satisfactory performance. Therefore, FedUAB, by incurring a computational and storage cost of $2Z$ times, realizes a model with superior performance and the capability for uncertainty assessment, which is acceptable in some scenarios.

The above provides a comprehensive discussion of the related work concerning FedUAB. Following this, we will proceed to outline the overview and specific details of FedUAB.

3. Overview

The architectural design of FedUAB has been briefly illustrated in the previous Fig. 1, combining the steps ① - ⑤ in this figure, the key components of FedUAB include:

- **Using Bayes by Backprop to training BNNs on clients.** This is the step ② in figure, how to train BNNs on clients with NonIID data, which differs from conventional centralized BNNs training and FL training, will be elaborated in detail in the following Section 4.2. This distributed variational inference training method brings an additional advantage, namely that random probability sampling can effectively protect the privacy of the samples, achieving the effect of sample-level differential privacy, this point is explained in Section 4.6.
- **Clients weights aggregation on server.** Step ④ is also distinct from the classical FedAvg algorithm, as each weight in FedUAB becomes a Gaussian distribution, we propose an innovative method for aggregating Gaussian distributions in Section 4.3.
- **Taking server global weights as prior on clients.** In step ② of Bayes by Backprop algorithm, every weight needs a prior. Unlike the centralized BNNs that frequently use a zero-mean Gaussian distribution as the prior, the global weights deployed in step ① of FedUAB serve as both the initial weights for training and the fixed prior throughout the clients training. The advantages it possesses, similar to those of the FedProx (Li et al., 2020), are described in Section 4.4.
- **Initial variance of weights.** FedUAB updates the training model through the repeated iteration of steps ① - ④, However, the initial weights, particularly the standard deviation σ, assigned during the inaugural round of global training, exert considerable influence on both model accuracy and uncertainty estimation, Section 4.5 is a theoretical analysis regarding the topic.
- **Decomposition of uncertainty.** One of the major advantages of FedUAB is that model has ability to quantify uncertainty of sample. Monte Carlo sampling can not only improve accuracy but also decompose the uncertainty into aleatoric and epistemic components, as introduced in ⑤, Section 4.7 presents the comprehensive principles, while Section 5.6 illustrates its functionality through specific sample cases.

The main mathematical notations and their explanations that appear in this paper are listed in Table 2.

4. Methods of feduab

This section provides a detailed explanation of the algorithm design of FedUAB based on the brief introduction in the previous section. It first outlines the fundamental principles of FL in Section 4.1, followed by a step-by-step introduction to the improvements of FedUAB, which demonstrate how to stabilize and efficiently implement BNNs in the FL environment.

4.1. Federated learning

FL addresses training challenges across a multitude of devices, orchestrated by a central server. In the context of a FL computing framework comprising a server denoted by S and a set of clients represented by C, the client population is quantified by N_c. In each round of global training, S selects K clients to participate training(although clients are chosen randomly, the actual selection process is often predicated on their computational and communicative capabilities at the time of selection). The client C_k has its local dataset D_k, the number

Table 2
Summary of notations.

Notation	Explanation	Used
S	FL server	
C	set of all FL clients	
N_c	number of all clients	
K	number of selected clients in FL training round	(1)(2)(7)(8)
C_k	kth client	
D_k	dataset of client k	
n_k	number of clients training samples in dataset D_k	(2)(7)(8)
n	sum of the number of clients training samples	(2)(7)(8)
ω	model weights	(1)
ω_k	local model weights of client k	(1)
ω_g	global model weights of server	(1)
$L_k(\omega)$	local training loss of client k	(1)
ω_k^t	local model weights of client k at round t	(2)
ω_g^t	global model weights of server at round t	(2)
$p(\omega\|D)$	true posterior of weights ω given data D	(3)
$Q(\omega\|D)$	variational posterior of weights ω given data D	(3)
θ	set of trainable parameters in variational posterior	(3)(4)(5)(6)
$Q(\omega\|\theta)$	variational posterior of weights ω given parameters θ	(4)(5)(6)(10)
$p(D\|\omega)$ $p(y\|x,\omega)$	likelihood of data D (x, y) given weights ω	(4)(5)(6) (11)(12)
$p(\omega)$	prior of weights ω	(4)(5)(6) (9)(10)
L	training loss of Bayes by Backprop	(5)(6)
L_E	likelihood loss	(5)(6)
L_C	complexity loss	(5)(6)
M	number of model weights	(6)(10)
$\omega_{k,m}^t$	m-th weight of client k model at round t	(6)
$\omega_{g,m}^t$	m-th weight of global model at round t	
Z	times of Monte Carlo sampling	(6)(10)(11)(12)
B	number of batches in client training	
L^b	training loss in b-th mini-batch	(10)
γ^b	hyperparameter of complexity loss in b-th mini-batch	(10)
μ	mean parameter of weights ω Gaussian distribution	(7)(8)
σ	standard deviation parameter of the weights ω Gaussian distribution	(7)(8)
ρ	parameter to ensure standard deviation σ are not negative	
H	hyperparameter for initial variance reduction	

of samples in dataset D_k is n_k (sum of the number of samples: $n = \sum_{k=1}^{K} n_k$), local model weights ω_k. The central server sustains a set of global model weights denoted as ω_g. The local loss for client C_k is given by $L_k(\omega)$. The objective of FL is to optimize the global model weights ω_g to reduce the collective loss across K clients (Yang et al., 2019).

$$\min_{\omega_1,\ldots,\omega_k} L(\omega) = \frac{1}{K}\sum_{k=1}^{K} L_k(\omega). \quad s.t. \quad \omega_1 = \omega_2 = \cdots = \omega_K = \omega_g \tag{1}$$

The FedAvg algorithm is widely recognized as the predominant method for weight aggregation in FL scenarios. In the training of FL at round t, Client k conducts model training on its local dataset D_k, yielding local model weights ω_k^t. Subsequently, the server amalgamates these weights from the participating clients. The resultant global model weights ω_g^{t+1}, as delineated in Eq. (2), are computed as a linear combination of the K clients' weights ω_k^t (McMahan et al., 2017):

$$\omega_g^{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \omega_k^t. \tag{2}$$

Our proposed Bayesian FL approach that builds upon the traditional FL and FedAvg algorithms mentioned above. Our approach corporates BNNs in the model weights and loss function. In the following section, we elaborate on the convergence of FL with BNNs.

4.2. Bayes by backprop for FL

This section focuses on addressing the challenge of training BNNs on FL clients. we use Bayes by Backprop (Shridhar et al., 2019), which employs variational inference to estimate the posterior distribution of the model weights. We will furnish an exhaustive account of its implementation on FL clients, complemented by a detailed exposition of the mathematical formulations underpinning the loss function.

In the Bayes by Backprop approach, the primary objective is the derivation of a robust posterior distribution. Since the true posterior $p(\omega|D)$ is typically intractable, the main task is to learn a variational posterior $Q_\theta(\omega|D)$, minimize the Kullback–Leibler(KL) divergence towards the actual posterior distribution $p(\omega|D)$, which is computed from trainable parameters θ, thus it is necessary to train the optimal

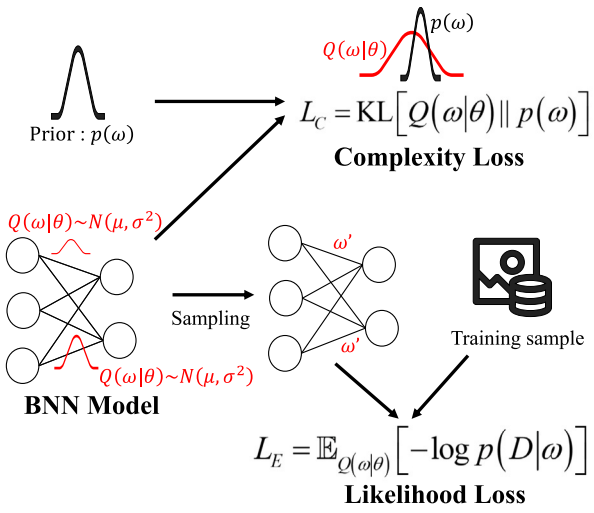

Fig. 3. The schematic of loss function in Bayes by Backprop algorithm. The loss function is bifurcated into two components: the likelihood loss derived from variational sampling inference and the complexity loss, which is predicated on the KL divergence. The key points of the two losses generated by the variational posterior Q are specifically marked in red.

parameters

$$\theta^* = \underset{\theta}{\operatorname{argmin}} KL\left[Q_\theta(\omega|D) \parallel p(\omega|D)\right]. \tag{3}$$

This derivation constitutes an optimization problem for Eq. (3), its solution has been clearly derived in many studies (Blundell et al., 2015; Graves, 2011), we omit the derivation steps and directly present the conclusion, it is expressed as the sum of two constituent terms:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}(\mathbb{E}_{Q(\omega|\theta)}\left[-\log p(D|\omega)\right] + KL\left[Q(\omega|\theta) \parallel p(\omega)\right]). \tag{4}$$

The aforementioned pertains to the loss function within the Bayes by Backprop algorithm, which is referred to as the free variational energy. It can be expressed as two components, as depicted in Eq. (5) :

$$\begin{aligned} L &= L_E + L_C \\ &= \mathbb{E}_{Q(\omega|\theta)}\left[-\log p(D|\omega)\right] + KL\left[Q(\omega|\theta) \parallel p(\omega)\right]. \end{aligned} \tag{5}$$

The first item $L_E = \mathbb{E}_{Q(\omega|\theta)}\left[-\log p(D|\omega)\right]$ is contingent upon the data set D and is designated as the likelihood loss. $p(D|\omega)$ is likelihood of data D given the current model weights ω, when the data D are represented as a set of inputs x and outputs y, $p(D|\omega)$ can also be expressed as $p(y|x,\omega)$. $-\log p(D|\omega)$ is negative log likelihood. $\mathbb{E}_{Q(\omega|\theta)}$ represents the negative log-likelihood needs to be expected based on the current variational posterior $Q(\omega|\theta)$.

The second item $L_C = \mathrm{KL}\left[Q(\omega|\theta) \parallel p(\omega)\right]$ is dependent on the prior $p(\omega)$, this measure represents the KL divergence between the current variational posterior and the prior distribution, thus called complexity loss. How to choose the prior $p(\omega)$ is a core issue that will be discussed in detail later.

According to the above analysis, it can be seen that variational free energy L encapsulates a balance between the model's adherence to the data D and its proximity to the prior distribution $p(\omega)$. In Fig. 3, the two components of the variational free energy loss function are displayed more clearly and explicitly. Next, we will explain how to use Monte Carlo sampling and the Gaussian distribution of the variational posterior to efficiently compute L, as well as updating parameters θ during the training process.

At first, the variational posterior $Q_\theta(\omega|D)$ utilize the simplest Gaussian distributions. In a more intuitive manner, it replaces each weight in the traditional DNN with a Gaussian distribution, using mathematical symbols can facilitate clearer expression as illustrated below: if the model has M weights, then variational posterior $Q(\omega^t_{k,m}|\theta^t_{k,m})$ of the mth weight of client k is can be represented as a Gaussian distribution: $\omega^t_{k,m} \sim N(\mu^t_{k,m}, {\sigma^t_{k,m}}^2)$. Because of $\sigma^t_{k,m}$ is always non-negative, we parameterize the standard deviation with ρ and Softplus function as $\sigma^t_{k,m} = log(1+exp(\rho^t_{k,m}))$. Therefore, in our proposed model, each weight has two parameters $\theta^t_{k,m} = (\mu^t_{k,m}, \rho^t_{k,m})$ that need to be trained, resulting in twice the number of parameters compared to traditional models.

Secondly, in each forward inference of the BNN model, pointwise samplings of the variational posterior $Q_\theta(\omega|D)$ are conducted, and the likelihood $p(D|\omega)$ are derived based on the sampled values. In client C_k, each weight $\omega^t_{k,m}$ is obtained by Gaussian sampling a unit Gaussian $N(0,1)$, shifting it by a mean $\mu^t_{k,m}$ and scaling by a standard deviation $\sigma^t_{k,m}$, local reparameterization trick (Kingma et al., 2015) can be used to accelerate the process. Then local model forword on its local dataset D_k at round t to obtain the likelihood $\hat{p}(D_k|\omega^t_k)$, and calculate the negative log-likelihood $-\log \hat{p}(D_k|\omega^t_k)$. Next, we employ Monte Carlo method to estimate the expectation, we can repeat mentioned above variational inference process Z times and calculate the average to obtain the likelihood loss L_E.

Thirdly, the computation of complexity loss L_C is relatively straightforward: the process solely entails the computation of the KL divergence between the variational posterior $Q(\omega^t_k|\theta^t_k)$ and the prior $p(\omega^t_k)$. The selection of prior distribution will be elaborated in the subsequent sections. For a model with M weights, each model weight requires individual computation, culminating in the aggregate sum.

In summary, the loss function L using Bayes by Backprop algorithm on FL clients training can be detailedly expressed by the following expression:

$$\begin{aligned} L &= L_E + L_C \\ &= \frac{1}{Z}\sum_{z=1}^{Z}\left[-\log \hat{p}_z\left(D_k|\omega^t_k\right)\right] \\ &+ \sum_{m=1}^{M} KL\left[Q\left(\omega^t_{k,m}|\theta^t_{k,m}\right) \parallel p\left(\omega^t_{k,m}\right)\right]. \end{aligned} \tag{6}$$

where Z is the sampling times of Monte Carlo method. Even if $Z = 1$, it can still allow the model to be trained relatively well (Shridhar et al., 2019), the performance improvement brought by larger Z will also be demonstrated in the subsequent experiments. And by combining other innovations in this study, we further optimize L in mini-batch form.

4.3. Weights aggregation

Weights aggregation of clients is required every communication round by the server in FL, in contrast to the traditional FL where FedAvg directly applies linear averaging, the weights of FedUAB are Gaussian distributions. The method for aggregating Gaussian distributions from different clients is discussed in this section.

In previous studies that combined the Bayes by Backprop with FL, these studies often directly imitate the FedAvg algorithm by linearly aggregating parameters μ and ρ, which can also achieve relatively good results (Zhang et al., 2022). However, such direct aggregation is not reasonable, especially for parameter ρ, which is introduced to ensure the non-negativity of standard deviation σ. The relationship between them is a non-linear softplus function $\sigma = Softplus(\rho) = \log(1+\exp(\rho))$. A simple analysis shows that, since the first-order derivative of the softplus function is always less than 1, linear aggregation of ρ will inevitably cause additional reduction in standard deviation σ.

The solution to this problem within the FedUAB framework enjoys stronger theoretical support. In each round of aggregation, we will aggregate the Gaussian distributions of weights $\omega^t_k \sim N(\mu^t_k, {\sigma^t_k}^2)$ from multiple clients to form a new distribution $\omega^{t+1}_g \sim N(\mu^{t+1}_g, {\sigma^{t+1}_g}^2)$, which should also remain a Gaussian distribution.

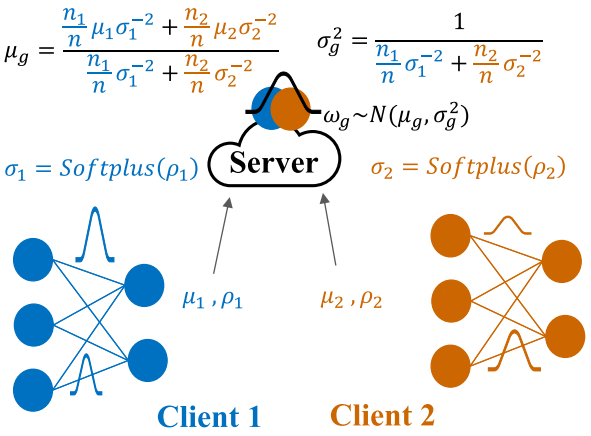

Fig. 4. A case of weights aggregation of FedUAB. This figure depicts the simplest scenario of weights aggregation involving only two clients. The relationship between the aggregated Gaussian distribution and the individual Gaussian distributions of the two clients is distinctly depicted through the use of two contrasting colors.

The most intuitive approach we can think of is the addition of independent Gaussian distributions, as shown in equation:

$$\mu_g^{t+1} = \sum_{k=1}^{K} \frac{n_k}{n}\mu_k^t, \ (\sigma_g^{t+1})^2 = \sum_{k=1}^{K} (\frac{n_k}{n}\sigma_k^t)^2. \tag{7}$$

However, the method of Eq. (7) will cause variance reduction of $\frac{1}{\sqrt{K}}$ in each round, and after several rounds the variance will quickly approach zero, it cannot be used in this study.

We ultimately discovered a suitable solution for aggregating BNN weights of FedUAB based on conflations of probability distributions, this approach serves as a mechanism for amalgamating a finite set of probability distributions (Genest & Zidek, 1986; Hill, 2011). The problem of conflation has been well studied, especially conflation of Gaussian distribution, combining FedAvg algorithm, the aggregation method is as following equation:

$$\mu_g^{t+1} = \frac{\sum_{k=1}^{K} \frac{n_k}{n}\mu_k^t(\sigma_k^t)^{-2}}{\sum_{k=1}^{K} \frac{n_k}{n}(\sigma_k^t)^{-2}}, \quad (\sigma_g^{t+1})^2 = \frac{1}{\sum_{k=1}^{K} \frac{n_k}{n}(\sigma_k^t)^{-2}}. \tag{8}$$

For a more intuitive illustration of the process in Eq. (8), Fig. 4 provides a simplified representation of the weights aggregation process between two clients in FedUAB framework.

It is evident that in the linear aggregation process, the reciprocals of variances, weighted by their squared values, act as the coefficients for different clients, with lower variances being associated with higher weights and vice versa. This aggregation method is suitable for evaluating diverse clients, avoiding unnecessary changes of weights, and its rationality and superior performance will be substantiated through experiments.

4.4. Global weights as prior

This paragraph addresses the issue of setting appropriate prior distributions for BNNs trained in clients of FedUAB. Setting the prior of BNNs is often typically a challenging task, poor predictive performance resulting from underfitting has impeded recent interest in learning large variational BNNs, because their performance is very sensitive to the prior over weights (Fortuin, 2022). Current practice often fixes the prior to standard values or tunes them using heuristics, the classic method involves using a zero-mean Gaussian prior, which serves to implement L_2 regularization (Shridhar et al., 2019).

In the server-clients structure of FL, we consider the prior of clients from a new perspective. In Bayesian frameworks, the prior probability encapsulates the likelihood of an event prior to the incorporation of new evidence, reflecting initial beliefs or assessments grounded in pre-existing knowledge or experience. For clients in FL, before each local training, the server aggregated global model are the latest and most accurate prior information for all clients of next round, so the global model weights serve as the prior of the client model weights, at round $t+1$ the mth weights of client k is distribution $\omega_{k,m}^t$, its prior is the latest global model weights broadcasted by the server in this communication round:

$$p\left(\omega_{k,m}^{t+1}\right) = N(\mu_{g,m}^t, (\sigma_{g,m}^t)^2) \tag{9}$$

This approach, which uses global weights as both the initial weights for the next round of the client and the prior distribution, inadvertently saves communication resources between the server and the clients. Imagine if the server had to send an additional set of prior to the clients, the communication volume could potentially double. The global weights prior have been attempted and proven to be convergent in previous research (Zhang et al., 2022), and they has also been found to have regularization effect, similar to FedProx, which mitigates statistical heterogeneity by aligning client-specific updates with the global model, thus attaining equilibrium between the central model and client models that may otherwise diverge.

Based on this prior and loss in Eq. (6), we will discuss the specific improvements to the loss L and hyperparameter settings when FL clients use mini-batch for training. The training of neural networks is usually in mini-batch mode, especially on FL clients where the data quantity, computing power, and memory capacity are often limited. As in Eq. (10), the data D_k on client k is divided into B equal size batches, during the training of the bth batch: the likelihood loss L_E^b is the mean negative log probability of this batch data D_{kb}, the complexity loss L_C^b of this batch needs to be multiplied by a hyperparameter γ^b to control the degree of regularization (Shridhar et al., 2019):

$$\begin{aligned} L^b &= L_E^b + L_C^b \\ &= \frac{1}{Z}\sum_{z=1}^{Z}[-\log \hat{p}_z\left(D_{kb}|\omega_k^t\right)] \\ &+\gamma^b \sum_{m=1}^{M} KL\left[Q\left(\omega_{k,m}^t|\theta_{k,m}^t\right) \parallel p\left(\omega_{k,m}^t\right)\right]. \end{aligned} \tag{10}$$

The setting of hyperparameter γ^b is a complex problem. In traditional Bayes by Backprop algorithm practice, it has been attempted to evenly distribute it to each batch($\gamma^b = \frac{1}{B}$), or to gradually decrease it during the training process($\gamma^b = \frac{2^{B-b}}{2^B-1}$) (Blundell et al., 2015). In FedProx, a comparable hyperparameter is instrumental, with various magnitudes tested across a confined set of candidates, such as{0.001, 0.01, 0.1}(Li et al., 2020).

According to previous experience, the hyperparameter γ^b serves as a trade-off between the benefits and drawbacks. A larger value of a indicates a greater need for clients to approximate the global model while suppressing NonIID, whereas a smaller value gives more emphasis on the personalized performance on the clients. In practical application, this parameter is set in accordance with the principles outlined above. In the experiments presented later in this paper, we have consistently employed a fixed value of $\gamma^b = 10^{-4}$, which has yielded satisfactory results.

4.5. Variance reduction

In this section, we provide an exhaustive examination of the initial variance in weights for the FedUAB model, examining its implications for the trade-off between model precision and uncertainty quantification capabilities: if the variance is too large, the output of the model varies greatly, resulting in a larger likelihood loss, but if the variance is too small, the BNNs will degenerate into traditional neural networks

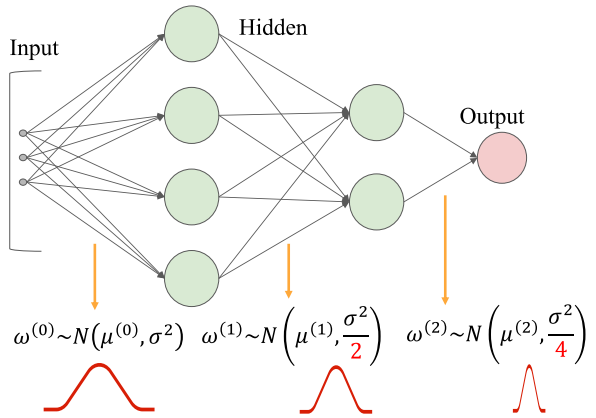

Fig. 5. Case of initial variance reduction in FedUAB. The initial variance of each layer's weights in the model is reduced by 1/2 compared to the previous layer.

and lose the ability of assessing uncertainty. Setting an appropriate initial variance value is the key to the success of BNNs model (Jospin et al., 2022).

Many papers related to BNNs have proposed that in multi-layer models, have been presented that the weight variances should be scaled according to the width of the network to avoid excessive accumulation of variance, the reason for this can be easily explained: whether it is a fully connected layer, convolutional layer, or other types of layers, the neuron's input represents a weighted sum of outputs from the preceding layer's neurons, determined by their associated weights ω. As a consequence of the Central Limit Theorem, the variance of the input of a neuron is also the linear sum of the variances (Neal & Neal, 1996). The accumulation of variances across layers leads to excessive uncertainties in the output of a BNNs, which makes it difficult to train the model to achieve high accuracy, this is also one of the reasons why BNNs are currently difficult to widely apply in large-scale deep learning (Goan & Fookes, 2020)

Therefore, in order to control the layer-by-layer accumulated variances in this study, it is essential to progressively decrease the variances of model weights at each layer. we decrease the initial variance of the weights by $1/H$ each layer, H becomes another hyperparameter. In the derivation of previous paper (Neal & Neal, 1996), let H denote the count of neurons in the preceding layer of a densely connected network that lacks a nonlinear activation function, this is evidently too large in practical use, as the weight variance of the subsequent layers approaches 0. Based on experiments and experience, the subsequent experiments in this study demonstrate the effect of $H = 2$, and a performance comparison is made with non-reduced variance ($H = 1$).

The schematic diagram of variance reduction is shown in Fig. 5. If first layer variance is 0.1^2 (assume $\sigma = 0.1$), and in the second layer it is $\frac{0.1^2}{2}$, and so on. The value of H should play the role of balancing the accuracy and assessing uncertainty, and it can be flexibly adjusted based on the model structure and depth in practice.

4.6. Sample level privacy protection

This section discusses the privacy preservation capabilities of FedUAB, model weights of FedUAB are Gaussian distributions instead of fixed values, making it naturally possess certain data privacy protection capabilities.

FL effectively safeguards client data privacy by preventing direct exposure to adversaries. However, adversaries may still infer private information through the analysis of the uploaded model weights or gradients, thereby attempting to reconstruct the original training data. Therefore, there have been many further studies on improving the security of FL. The prevalent technique in this domain is DP, which incorporates the addition of noise to parameters prior to aggregation, adding noise to clients or each of samples (Abadi et al., 2016) will inevitably lead to a decrease in model performance. Balancing between enhancing security and model performance is a trade-off. Minimizing the impact of added noise on model accuracy is a good solution, which has been attempted from the perspective of interpretability in deep learning (Li et al., 2023) and Bayesian privacy accounting method (Triastcyn & Faltings, 2019).

FedUAB is also a way to add controlled noise to the training data. The model samples all weights every time it performs a forward calculation, and the sampling process is close to adding noise, where the amount of noise depends on the standard deviation σ of the weights, the sampling is completely random and indeed achieves the effect of adding random noise to the weights. Moreover, the standard deviation σ is learned through data training, and its impact on model accuracy is controllable. As demonstrated by subsequent experiments, the model accuracy trained by FedUAB even surpasses that of FedAvg. In a word, FedUAB can achieve privacy protection similar to DP and can achieve privacy protection at the sample level.

4.7. Quantification and decomposition of uncertainty

We will specifically explain how to decomposite the uncertainty of a given data $D^* = \{x^*, y^*\}$, where x^* represents the data sample and y^* denotes its corresponding predicted label within a classification context. $p(D^*|\omega)$ is the predicted probability of the model's output on the data D^* given the current model weights ω, the calculation of this probability is exactly the same as the process for calculating likelihood in the previous equation (5). To compute its expectation we needs to use the Monte Carlo method, sample Z times weights ω and calculate the average to obtain its approximate value, and the variance of predicted probability $p(D^*|\omega)$ is the uncertainty of the output and can be decomposed into two parts (Kwon et al., 2018):

$$E\left[p(D^*|\omega)\right] \approx \bar{p}(y^*|x^*) = \frac{1}{Z}\sum_{z=1}^{Z}\hat{p}_z(y^*|x^*), \tag{11}$$

$$\begin{aligned}\mathrm{Var}\left[p(D^*|\omega)\right] = &\underbrace{\frac{1}{Z}\sum_{z=1}^{Z}\mathrm{diag}\left(\hat{p}_z\right) - \hat{p}_z\hat{p}_z^T}_{\text{aleatoric}} \\ &+ \underbrace{\frac{1}{Z}\sum_{z=1}^{T}\left(\hat{p}_z - \bar{p}\right)\left(\hat{p}_z - \bar{p}\right)^T}_{\text{epistemic}}.\end{aligned} \tag{12}$$

Just as indicated in the above Eq. (12), distinguishing the predictive variance's uncertainty into aleatoric and epistemic elements is crucial, as it enables the modeler to evaluate the scope for enhancing model performance: while aleatoric uncertainty arises from model limitations and can be mitigated by model refinement, epistemic uncertainty quantifies data variability. OoD points are characterized by elevated epistemic uncertainty, indicating a need for cautious prediction rather than inaccurate assumptions (Der Kiureghian & Ditlevsen, 2009). Consequently, a modeler can discern whether data quality is suboptimal, indicated by elevated epistemic uncertainty, or if model deficiencies underlie poor performance, signified by high aleatoric uncertainty. The former may be addressed by data augmentation, while the latter necessitates model enhancement (Jospin et al., 2022).

Quantification and decomposition of uncertainty are significant contributions of FedUAB. They will undoubtedly play an important role in scenarios such as autonomous driving and engineering control. They can also contribute to optimizing the performance of training during the model's training process (Depeweg et al., 2018). We will also visually demonstrate its effectiveness through images in experiments.

4.8. Summary of FedUAB

Combined with what was mentioned above, we summarize our FedUAB algorithm in this section, a pseudo-code for FedUAB is shown in Algorithm 1.

Algorithm 1 Uncertainty-Aware BNNs for FL (FedUAB)

Input: FL server: S; Set of all clients: C; Number of clients to be selected in each round: K, Dataset of client k: D_k; Initial model weights(with decreasing initial variance): w_g^0.
Output: Trained global model weights w_g^T.

while The training of the global model remains unfinished **do**
  An round t, FL server S selects K clients from all FL clients C to participate training, S The server disseminates the most recent global model weights w_g^t to all participated clients C_K.
  for all Selected client $c_k \in C_K$ **do**
    FL client c_k takes the received global weights as the initial weights and prior (as equation (9)) for local training, and trains model on local data D_k using Bayes by Backprop algorithm, with the loss function as shown in equation (10).
    FL client c_k sends local trained model weights w_k^t to FL server S.
  end for
  FL server S aggregated all the received model weights by equation (8), gets the new global model weights w_g^{t+1}
  if The global model training is concluded upon achieving a sufficiently low training loss or an adequately high test accuracy. **then**
    Return trained global model weights w_g^T.
  end if
end while

5. Experiments

Experiments conducted on three public datasets have substantiated the performance advantages of FedUAB. The validation encompassed the performance of both the server-side global model and the client-side personalized models.

5.1. Baselines

FedUAB needs to be compared with traditional FL, we select the classical **FedAvg** and previously mentioned **FedProx** for comparison.

Next, the impact of the Monte Carlo sampling time Z in Eq. (6) of the model also needs to be experimentally verified. we compare the effects of different sampling numbers ($Z = 1, 2, 5$) on model training and inference, we abbreviate them as **FedUAB-1**, **FedUAB-2**, and **FedUAB-5**.

Additionally, among the many baselines mentioned in the previous Section 2 on related work, we have selected **pFedBayes** and **BPFed**, which are most closely related to our work and also implement BNNs via variational inference on the clients in FL.

Furthermore, the effectiveness of several major innovative points, including weights aggregation, prior distribution, and variance reduction, is verified through ablation experiments.

5.2. Performance metrics

This paper aims to evaluate the effectiveness of the newly proposed FedUAB by comparing its performance against the aforementioned existing baselines in FL. Different baselines are implemented using the same network architecture, client data, and training participation rates to ensure a fair comparison.

The primary performance metrics include the convergence rate of the trained models, assessed through accuracy and log-likelihood loss on the test set following each round of global training, as well as the highest accuracy attainable by the models.

The models requiring performance evaluation include the **global** model of server and the **personalized** models of clients. The validation of the global model is conducted on the entire test dataset. In parallel, given the focus of many analogous studies on personalized FL, we adopt their approach for testing client-side personalized models. This involves each client validating the accuracy and loss on test data that has the same distribution as their training data, specifically matching label distributions, to standardize comparisons, we provide the mean personalized test metrics across all clients.

Table 3
Datasets.

Dataset	Type	Categories	Training set	Test set
MNIST	Image	10	60 000	10 000
CIFAR-10	Image	10	50 000	10 000
CNews	Text	10	50 000	10 000

Table 4
Models.

Dataset	Model structure	Weights
MNIST	input->hidden layer(50 units)-> hidden layer(50 units)->output	42 310
CIFAR-10	input-> conv(3 × 3 kernel, 32 channels) ->2 × 2 maxpooling -> conv(3 × 3 kernel, 32 channels) ->2 × 2 maxpooling -> conv(3 × 3 kernel, 32 channels) ->hidden layer(64 units) ->output	52 874
CNews	input->embedding(50000 to 16) ->conv(2 × 16 kernel, 10 channels), conv(3 × 16 kernel, 10 channels), conv(4 × 16 kernel, 10 channels) maxpooling-> linear(30 to 10)->output	801 780

5.3. Experimental setting

Datasets. We generate the datasets of FL clients based on three public benchmark datasets: MNIST, CIFAR-10 and CNews. The first two datasets are well-known image recognition tasks, while the last one is a subset of the THUCNews dataset (Sun et al., 2016) for text classification tasks. Table 3 presents the fundamental characteristics of the three datasets.

Clients Setting. In order to thoroughly simulate the FL environment, we have established 100 clients with identical data volumes. In each round of global training, a random subset of 10 clients is selected to contribute to the training regimen, culminating in the aggregation of their respective model weights. Each client has a mere 10% chance of engaging in training per round. We posit that this FL setting, characterized by a multitude of clients and a low probability of participation, more accurately approximates real-world applications such as those found in the Internet of Things, thereby providing a more stringent test of the robustness of FL algorithms.

Data distribution on clients is a central issue in FL environments. We have established clients with both identically distributed(IID) and NonIID. IID stands for the scenario where each client uniformly assigns training data from the training set.

For NonIID, we simulate FL environment with label distribution skew, the datasets are partitioned based on labels, the disparity in label distribution among FL clients is modeled using a Dirichlet distribution with concentration parameter η, a lower value of η indicates greater data heterogeneity. Each client's label distribution is characterized by η values drawn from a continuous uniform distribution within the range $(0, \eta_{max}]$, we choose $\eta_{max} = 0.5$ (Wang et al., 2021). Furthermore, to

Table 5
Scenarios of clients data setting.

Scenarios	Distribution	Volume
All-IID	Uniform	600 or 500
All-Dirichlet	Dirichlet	600 or 500
All-Only2	Only two labels	600 or 500
Small-IID	Uniform	50
Small-Dirichlet	Dirichlet	50
Small-Only2	Only two labels	50

more effectively simulate a personalized FL scenario, an extreme non-identically distributed setting has been introduced where each client possesses data with only two labels, this method was employed in the original paper proposing FedAvg (McMahan et al., 2017).

The volume of training data for each client is also set under three scenarios. Firstly, all the training data are distributed evenly among each client, resulting in $60000/100 = 600$ samples (for MNIST) or $50000/100 = 500$ samples(for CIFAR-10 and CNews) per client. Secondly, in order to validate the data-efficient and anti-overfitting capability of FedUAB, each client is provided with only 50 samples.

Considering distribution and volume of clients data, each dataset should be divided into six experimental scenarios, as shown in Table 5, for comparative analysis.

Model. The main objective of this study is to validate the feasibility and effectiveness of combining BNNs with FL in the context of FedUAB, rather than maximizing model accuracy. Therefore, we employ small models with simple structures and strong universality on all three datasets, which nevertheless include commonly used network components such as fully connected layers, convolutional layers, and embedding layers. In addition, simple and low-parameter models are also suitable for scenarios with limited computational and communication resources in FL.

For MNIST digit recognition task, we utilized a straightforward multilayer perceptron comprising two hidden layers for our experiments. Additionally, a convolutional neural network(CNN) with three layers was applied to analyze the CIFAR-10 dataset. For the last CNews text classification task, we adopted a CNN proposed in paper (Kim, 2014), but it is necessary to simplify the model in order to adapt to the FL environment and improve the speed of simulation experiments, the dimension value of the embedding has been reduced to 16, the embedded words are processed by three parallel convolutional layers to extract textual features, culminating in the output through a linear layer. The brief structure and weight quantity information of the models used in the above three datasets are summarized in Table 4.

Hyperparameters Setting. FL has more hyperparameters to set than centralized deep learning, and many hyperparameter settings are limited by client resources. To ensure equitable comparison and underscore the general applicability of FedUAB, Uniform hyperparameter configurations were also maintained across all clients and algorithms.

In our experiments, 100 clients of FL are simulated, and during each round of training, a subset of 10 clients is randomly selected for engagement in the training and aggregation phase. The hyperparameter settings for clients in the FL training process are as follows: batch size $B = 10$, default SGD optimizer with a learning rate $r = 0.05$, local training epochs $E = 5$. As mentioned earlier, the coefficient of complexity loss in Eq (10) is $\gamma^b = 10^{-4}$.

Machines and Libraries. We simulate the FL setup (1 server and N devices) on a commodity machine with NVIDIA 3090 GPUs. All code is implemented in PyTorch Version 2.0.0.

5.4. Global model performance

In accordance with the previously mentioned setup, we conducted 2000 rounds of FL global training across all six data distribution scenarios on three datasets. We compared the accuracy and negative log-likelihood loss of the global model aggregated by the server on the full test set for all baselines(excluding BPFed, which lacks a global model) with FedUAB.

Experiments were conducted following the client data distribution partitions detailed in Table 5. The results from the all training datasets across three datasets are exhibited in Table 6, whereas Table 7 illustrates the outcomes under scenarios with small client data, specifically with each client possessing 50 samples.

Table 6 illustrates that across three disparate datasets with all data training, the FedUAB algorithm consistently demonstrated its anticipated superiority. The globally trained model by FedUAB outperformed the traditional FL algorithms FedAvg and FedProx in both accuracy and loss. Notably, as the number of MC sampling iterations, denoted by Z, increased, the performance of FedUAB improved. With $Z = 5$, the performance of FedUAB was already sufficiently optimal testing with $Z = 10$ showed that further performance gains were not as pronounced.

Given its close resemblance to pFedBayes, which can also be considered an ablation study of FedUAB without the novel parameter aggregation method and variance reduction techniques proposed in this paper, pFedBayes shows performance comparable to FedUAB on the MNIST and CNews datasets. This similarity is attributed to the shallow model depth employed (only three layers), where the advantage of variance reduction in FedUAB is not pronounced. However, on the CNews dataset with a deeper convolutional network consisting of five layers, FedUAB demonstrates a significant advantage over pFedBayes.

Furthermore, it can be observed that FedUAB exhibits greater advantages under NonIID client data scenarios, which is attributed to the global model serving as a prior distribution that mitigates the effects of client drift.

Subsequently, we analyze the performance in the small data client scenarios presented in Table 7, where one of the advantages of BNNs is their data efficiency.

It is evident that the model trained by FedUAB under conditions of limited data still retains its advantages, particularly with a significant lead on the CNews dataset. Although it may not always achieve the highest accuracy in certain scenarios, it consistently exhibits the lowest loss values. This is because traditional FL is prone to overfitting on clients with as few as 50 samples, leading to overly confident models with high negative log-likelihood loss. In contrast, FedUAB, based on BNNs, balances data efficiency with robustness against overfitting.

To illustrate this point more vividly, Fig. 6 presents the performance curves of the global model under the Small-Dirichlet data distribution scenario across three datasets, showing the changes in the global model's performance after each round of aggregation. It is evident that as training progresses, traditional frequentist FL methods (FedAvg and FedProx) cease to make progress in terms of test accuracy while their loss continues to increase, indicating overfitting. However, FedUAB effectively suppresses this overfitting, maintaining training loss without significant escalation, even if accuracy is not surpassing in some experiments. Compared to pFedBayes, FedUAB's variance reduction on model weights enhances both precision and training loss.

The advantages of FedUAB demonstrate robustness in practical applications. Given the high degree of non-identically distributed data and the limited data availability on clients, compounded by privacy protection and data scarcity, it is challenging for FL to implement early stopping mechanisms to find an appropriate end to training. The performance and anti-overfitting capabilities of FedUAB endow it with the robustness to adapt to various complex FL environments.

5.5. Personalized model performance

We analyze and compare the performance of clients personalized models between FedUAB and other baselines. As previously mentioned, personalized performance refers to the accuracy and loss of a client's model when evaluated on test data with the same distribution as its training data. In scenarios where data is IID, personalized testing is

Table 6
Global model performance on ALL training data.

Dataset	Method	All-IID		All-Dirichlet		All-Only2	
		Acc(%)	Loss	Acc(%)	Loss	Acc(%)	Loss
MNIST	FedAvg	97.68	0.163	97.35	0.171	97.07	0.155
	FedProx	97.62	0.128	97.39	0.126	96.98	0.134
	pFedBayes	97.70	0.126	97.44	0.114	96.93	0.115
	FedUAB-1	97.52	0.189	97.33	0.178	96.33	0.134
	FedUAB-2	97.79	0.153	97.55	0.122	96.89	0.124
	FedUAB-5	**97.84**	**0.114**	**97.66**	**0.095**	**97.08**	**0.101**
CIFAR-10	FedAvg	77.51	0.706	72.44	0.887	57.60	1.385
	FedProx	77.81	0.699	72.91	0.887	57.66	1.526
	pFedBayes	74.45	0.764	68.12	0.915	59.07	1.143
	FedUAB-1	76.11	0.745	70.31	0.930	51.48	1.832
	FedUAB-2	77.55	0.693	72.58	0.841	52.10	1.729
	FedUAB-5	**78.95**	**0.641**	**73.50**	**0.800**	**61.86**	**1.128**
CNews	FedAvg	88.20	0.732	86.34	0.949	81.02	1.001
	FedProx	90.06	0.628	85.03	1.067	83.59	0.872
	pFedBayes	**93.68**	**0.310**	91.10	0.452	89.49	0.398
	FedUAB-1	93.19	0.520	90.26	0.617	90.59	0.433
	FedUAB-2	93.50	0.421	91.07	0.476	89.40	0.426
	FedUAB-5	93.23	0.380	**91.25**	**0.428**	**90.40**	**0.376**

Table 7
Global model performance on small training data.

Dataset	Method	Small-IID		Small-Dirichlet		Small-Only2	
		Acc(%)	Loss	Acc(%)	Loss	Acc(%)	Loss
MNIST	FedAvg	93.89	0.362	92.37	0.435	92.87	0.417
	FedProx	93.36	0.368	93.29	0.377	93.07	0.378
	pFedBayes	**94.95**	0.243	94.62	0.258	**94.41**	0.253
	FedUAB-1	94.09	0.399	93.34	0.420	92.84	0.395
	FedUAB-2	94.51	0.320	94.23	0.309	93.59	0.301
	FedUAB-5	94.88	**0.240**	**94.68**	**0.246**	94.33	**0.237**
CIFAR-10	FedAvg	57.31	1.824	**56.72**	1.802	53.89	1.858
	FedProx	55.92	1.837	55.82	1.881	53.57	1.811
	pFedBayes	56.72	1.498	45.96	1.576	47.70	1.497
	FedUAB-1	56.83	1.663	52.21	1.798	49.79	1.610
	FedUAB-2	57.39	1.525	54.01	1.598	49.83	1.564
	FedUAB-5	**58.88**	**1.451**	55.13	**1.539**	**54.22**	**1.374**
CNews	FedAvg	71.58	2.215	73.13	1.681	70.51	1.479
	FedProx	76.64	1.776	74.47	1.665	72.76	1.581
	pFedBayes	82.76	0.896	80.26	0.892	80.08	0.793
	FedUAB-1	82.41	1.054	81.28	0.964	78.69	1.031
	FedUAB-2	83.38	0.875	80.75	0.926	81.34	0.930
	FedUAB-5	**84.44**	**0.801**	**81.70**	**0.787**	**82.20**	**0.696**

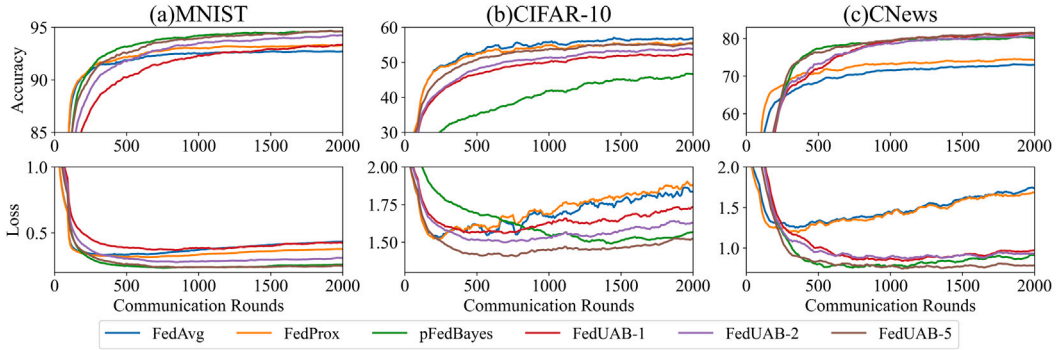

Fig. 6. The global model test accuracy and loss across three datasets under the Small-Dirichlet scenario. FedUAB achieved a high accuracy and effectively mitigated overfitting.

equivalent to global model testing; hence, we focus on NonIID situations, especially highly personalized scenarios where clients have data from only two classes. Table 8 presents the specific results, the results represent the average outcomes across all 100 participating clients on their personalized test datasets.

It can be observed from Table 8 that after fine-tuning on local clients, personalized models generally achieve higher accuracy on test data with the same distribution. FedUAB continues to demonstrate its superiority, even outperforming BPFed, which specifically emphasizes personalized FL. Fig. 7 illustrates the evolution of personalized performance in the Small-Only2 scenario.

It is evident that FedUAB outperforms other baselines in terms of personalized performance on client-side, with others exhibiting signs of overfitting. Notably, BPFed, which emphasizes personalization, partitions model parameters into shared and client-specific personalized factors, allowing each client to retain its own parameters, even with

Table 8
Personalized model performance.

Dataset	Method	ALL-Dirichlet		ALL-Only2		Small-Dirichlet		Small-Only2	
		Acc(%)	Loss	Acc(%)	Loss	Acc(%)	Loss	Acc(%)	Loss
MNIST	FedAvg	97.85	0.142	98.56	0.078	93.81	0.38	93.97	0.345
	FedProx	97.89	0.103	98.30	0.072	94.11	0.33	93.98	0.321
	pFedBayes	98.32	0.075	99.07	0.032	95.65	0.211	96.42	0.162
	BPFed	96.66	0.571	99.16	0.045	90.60	0.931	**97.37**	0.127
	FedUAB-1	98.18	0.124	98.69	0.066	94.65	0.349	95.83	0.234
	FedUAB-2	98.41	0.087	99.12	0.039	95.49	0.243	96.48	0.165
	FedUAB-5	**98.51**	**0.060**	**99.21**	**0.027**	**95.90**	**0.193**	96.96	**0.122**
CIFAR-10	FedAvg	84.26	0.568	93.01	0.242	74.03	1.298	86.42	0.564
	FedProx	84.51	0.576	92.74	0.241	72.67	1.314	86.46	0.521
	pFedBayes	83.31	0.515	91.12	0.249	70.82	0.952	85.13	0.476
	BPFed	82.52	0.689	90.58	0.232	63.86	2.198	77.45	1.175
	FedUAB-1	83.78	0.555	91.15	0.261	72.18	1.247	85.04	0.597
	FedUAB-2	84.82	0.490	92.20	0.221	74.12	1.124	85.41	0.522
	FedUAB-5	**85.54**	**0.453**	**92.77**	**0.199**	**74.95**	**0.999**	**86.61**	**0.433**
CNews	FedAvg	93.23	0.430	95.82	0.182	77.74	1.398	85.87	0.661
	FedProx	92.81	0.453	95.72	0.194	78.73	1.408	85.45	0.704
	pFedBayes	95.09	0.237	96.95	0.115	86.31	0.572	**91.56**	**0.302**
	BPFed	92.18	0.821	96.44	0.170	76.94	2.29	86.69	0.748
	FedUAB-1	94.27	0.349	96.98	0.119	86.48	0.699	88.23	0.526
	FedUAB-2	95.02	0.258	**97.02**	0.116	86.31	0.648	88.70	0.504
	FedUAB-5	**95.19**	**0.225**	97.00	**0.112**	**87.07**	**0.528**	91.47	0.302

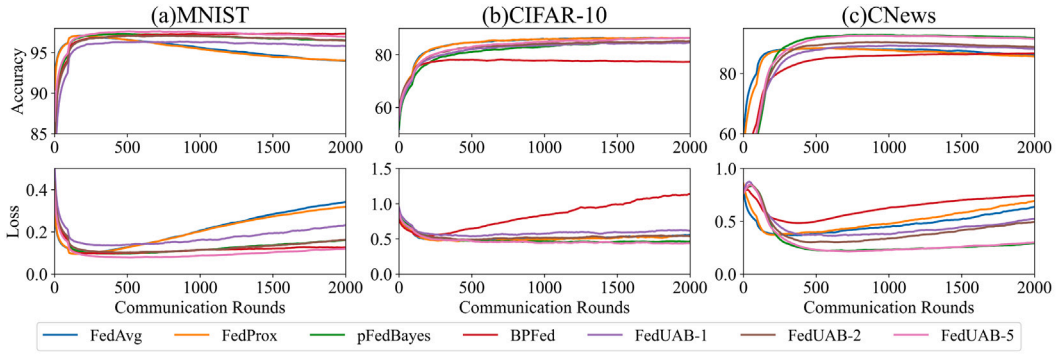

Fig. 7. The personalized model test accuracy and loss across three datasets under the Small-Only2 scenario. FedUAB demonstrates superior performance on client-side personalized models.

such personalized FL algorithms, our FedUAB has managed to outperform them in terms of personalized performance. we attribute this to the more realistic experimental environment setup employed in this paper, where only 10 out of 100 clients are randomly selected to participate in each training round, with a participation probability of just 10%. Under such conditions, it is challenging for clients to train highly effective personalized model parameters.

This effectively demonstrates the robustness of our FedUAB algorithm. Even with a limited number of clients involved in training (a common scenario in real-world FL applications where it is not guaranteed that every client will participate in every round), FedUAB maintains good model performance. After a simple round of fine-tuning, the personalized performance of the clients can still adapt to their individual data distributions.

5.6. Case of uncertainty decomposition

In the previous Section 4.7, the theory related to the uncertainty decomposition of FedUAB was explained in detail. In practical applications, this feature also represents the most distinctive advantage of FedUAB over traditional FL methods. In this section, we combine several typical actual samples to fully illustrate the practical significance of calculating the aleatoric uncertainty and epistemic uncertainty.

We utilize the model that was trained using FedUAB algorithm on the CIFAR-10 dataset from the previous experiments (FedUAB-5 of All-Dirichlet), this is primarily due to the more intuitive and visually appealing nature of the images within the CIFAR-10 dataset, as compared to the MNIST and CNews datasets, resulting in a superior presentation. The model yielded different uncertainties for correct and incorrect inferences, as well as OoD samples, providing a vivid demonstration when combined with the image of the samples.

In Fig. 8, the outputs and uncertainties on four test samples of CIFAR-10 dataset are presented respectively. In the two images above, both correctly identified as deer, but the difference of uncertainty between them is intriguing. The first image shows a very distinct deer with prominent antlers, highlighting its features, both types of uncertainties are very low. In contrast, the second image is not as easy to recognize, although it is correctly identified as a deer, the uncertainty is higher: the elevated aleatoric uncertainty indicates lower likelihood probabilities during model inference, as it can be easily misclassified as a horse, the higher epistemic uncertainty suggests significant variations in likelihood probabilities formed by the model during multiple variational sampling, this output of model is not very 'confident'.

The two figures below demonstrate that the model incorrectly confuses airplane and bird: their aleatoric uncertainties are high, indicating low likelihood probabilities, and the epistemic uncertainties are not low either. Despite making an error, the model exhibits a 'humble' behavior by demonstrating a higher level of uncertainty.

The next step involves a more challenging task of evaluating the OoD detection capability of the FedUAB model, let this model to

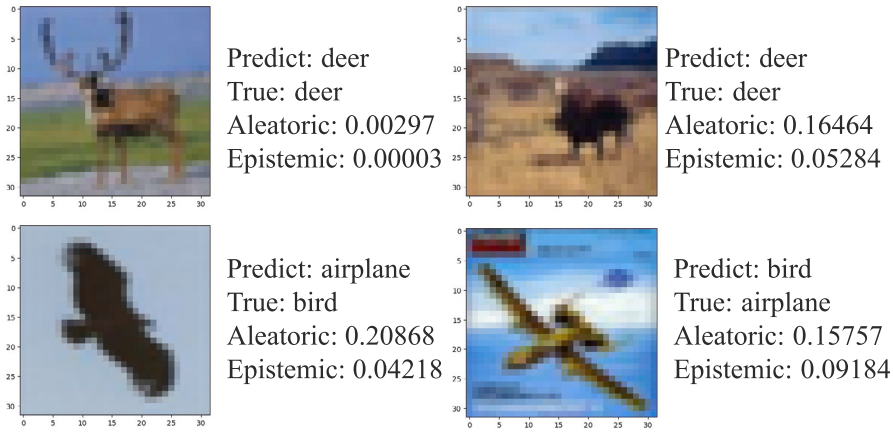

Fig. 8. Uncertainty decomposition on CIFAR-10 test samples. The cases of correct and wrong predictions are showcased. The first panel in the top left that predicts correctly with very little uncertainty is the best model performance, but for the others, it is also valuable for the model to quantify its lack of confidence.

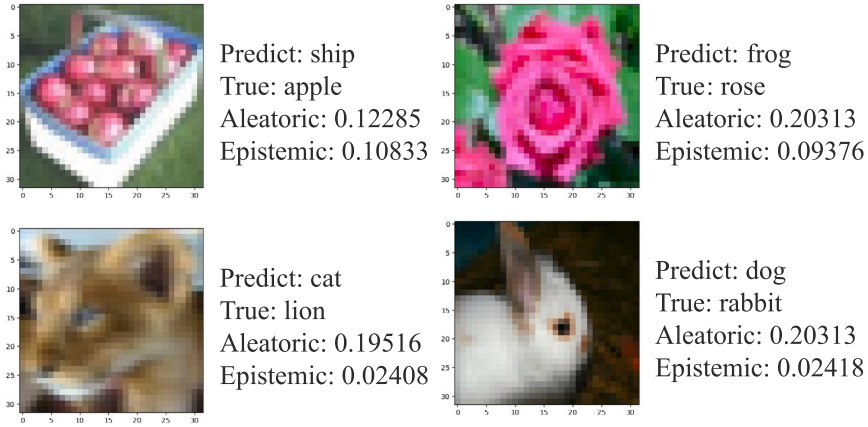

Fig. 9. Uncertainty Decomposition on OoD samples. The higher the uncertainty (especially epistemic uncertainty), the more unfamiliar samples are to the model, and the more likely they are OoD.

recognize the images from the CIFAR-100 dataset belonging to 'unseen' classes that it has not been exposed to before, the four selected examples in Fig. 9 are as follows. In the upper images, the epistemic uncertainty are significantly higher, the model exhibits low confidence in its predictive outcomes, suspecting that the samples may be OoD. The two below images illustrate examples of OoD samples with relatively low epistemic uncertainties, where it is evident that they are indeed close to the types they were misclassified as, and it is reasonable to have lower uncertainties than upper examples.

The presentation of the above figures further illustrates the specific roles of two types of uncertainties. Aleatoric uncertainty indicates the model's limited understanding, whereas epistemic uncertainty signifies the model's lack of familiarity with the sample data. The model trained by the FedUAB algorithm has the ability to quantitatively calculate these uncertainties, which will be of great practical value in engineering practice.

6. Conclusion

In summary, the FedUAB proposed in this paper is a practical approach within the field of BFL. The proposed method addresses several critical aspects of BFL, including client training, weight aggregation, selection of prior distributions, and management of parameter variances. And FedUAB quantifies sample uncertainty and ensures sample-level privacy through random sampling.

Experiments on global and personalized model testing across six distinct data distribution scenarios on three datasets demonstrate that, compared to traditional FL and prevalent BFL methods, FedUAB enhances test accuracy and reduces loss, mitigates client drift and overfitting, and exhibits robustness.

FedUAB introduces a higher computational complexity and double the communication cost compared to traditional FL, yet these are not prohibitively high when compared to most BFL methods. We believe that FedUAB represents a positive and effective endeavor in the field of BFL, holding potential for various practical applications across diverse scenarios.

CRediT authorship contribution statement

Pengfei Li: Writing – review & editing, Writing – original draft, Software, Methodology, Conceptualization. **Qinghua Hu:** Supervision, Resources, Project administration. **Xiaofei Wang:** Supervision, Methodology, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Qinghua Hu reports financial support was provided by National Natural Science Foundation of China. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We have uploaded the source code in https://github.com/lpf111222/FedUAB/.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security* (pp. 308–318).

Achituve, I., Shamsian, A., Navon, A., Chechik, G., & Fetaya, E. (2021). Personalized federated learning with gaussian processes. *Advances in Neural Information Processing Systems*, *34*, 8392–8406.

Al-Shedivat, M., Gillenwater, J., Xing, E., & Rostamizadeh, A. (2020). Federated learning via posterior averaging: A new perspective and practical algorithms. arXiv preprint arXiv:2010.05273.

Bhatt, S., Gupta, A., & Rai, P. (2024). Federated learning with uncertainty via distilled predictive distributions. In *Asian conference on machine learning* (pp. 153–168). PMLR.

Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, *112*(518), 859–877.

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural network. In *International conference on machine learning* (pp. 1613–1622). PMLR.

Cao, L., Chen, H., Fan, X., Gama, J., Ong, Y. S., & Kumar, V. (2023). Bayesian federated learning: A survey. arXiv preprint arXiv:2304.13267.

Chandra, R., & Simmons, J. (2024). Bayesian neural networks via MCMC: a python-based tutorial. *IEEE Access*.

Chen, H. Y., & Chao, W. L. (2020). Fedbe: Making bayesian model ensemble applicable to federated learning. arXiv preprint arXiv:2009.01974.

Chen, H., Liu, H., Cao, L., & Zhang, T. (2023). Bayesian personalized federated learning with shared and personalized uncertainty representations. arXiv preprint arXiv:2309.15499.

Depeweg, S., Hernandez-Lobato, J. M., Doshi-Velez, F., & Udluft, S. (2018). Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning. In *International conference on machine learning* (pp. 1184–1193). PMLR.

Der Kiureghian, A., & Ditlevsen, O. (2009). Aleatory or epistemic? Does it matter? *Structural Safety*, *31*(2), 105–112.

Du, J., Li, W., Liu, P., Vong, C. M., You, Y., Lei, B., & Wang, T. (2024). Federated learning using model projection for multi-center disease diagnosis with non-IID data. *Neural Networks*, Article 106409.

Fortuin, V. (2022). Priors in bayesian deep learning: A review. *International Statistical Review*, *90*(3), 563–591.

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

Genest, C., & Zidek, J. V. (1986). Combining probability distributions: A critique and an annotated bibliography. *Statistical Science*, 114–135.

Goan, E., & Fookes, C. (2020). Bayesian neural networks: An introduction and survey. In *Case studies in applied bayesian data science: CIRM Jean-Morlet chair, fall 2018* (pp. 45–87). Springer.

Graves, A. (2011). Practical variational inference for neural networks. *Advances in Neural Information Processing Systems*, *24*.

Hastings, W. K. (1970). *Monte Carlo Sampling Methods Using Markov Chains and Their Applications*. Oxford University Press.

Hernández-Lobato, J. M., & Adams, R. (2015). Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International conference on machine learning* (pp. 1861–1869). PMLR.

Hill, T. (2011). Conflations of probability distributions. *Transactions of the American Mathematical Society*, *363*(6), 3351–3372.

Jing, C., Huang, Y., Zhuang, Y., Sun, L., Xiao, Z., Huang, Y., & Ding, X. (2023). Exploring personalization via federated representation learning on non-IID data. *Neural Networks*, *163*, 354–366.

Jospin, L. V., Laga, H., Boussaid, F., Buntine, W., & Bennamoun, M. (2022). Hands-on Bayesian neural networks—A tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, *17*(2), 29–48.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 conference on empirical methods in natural language processing*. Association for ComputationalLinguistics.

Kingma, D. P., Salimans, T., & Welling, M. (2015). Variational dropout and the local reparameterization trick. *Advances in Neural Information Processing Systems*, *28*.

Kotelevskii, N., Vono, M., Durmus, A., & Moulines, E. (2022). Fedpop: A bayesian approach for personalised federated learning. *Advances in Neural Information Processing Systems*, *35*, 8687–8701.

Kwon, Y., Won, J. H., Kim, B. J., & Paik, M. C. (2018). Uncertainty quantification using bayesian neural networks in classification: Application to ischemic stroke lesion segmentation. *Medical Imaging with Deep Learning*, *4*(2).

Li, Z., Chen, H., Ni, Z., & Shao, H. (2023). Balancing privacy protection and interpretability in federated learning. arXiv preprint arXiv:2302.08044.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020). Federated optimization in heterogeneous networks. In *Proceedings of machine learning and systems, vol. 2* (pp. 429–450).

Li, X. C., & Zhan, D. C. (2021). Fedrs: Federated learning with restricted softmax for label distribution non-iid data. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining* (pp. 995–1005).

Lindinger, J., Rakitsch, B., & Lippert, C. (2022). Laplace approximated Gaussian process state-space models. In *Uncertainty in artificial intelligence* (pp. 1199–1209). PMLR.

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

Neal, R. M., & Neal, R. M. (1996). Priors for infinite networks. In *Bayesian learning for neural networks* (pp. 29–53). Springer.

Pearce, T., Leibfried, F., & Brintrup, A. (2020). Uncertainty in neural networks: Approximately bayesian ensembling. In *International conference on artificial intelligence and statistics* (pp. 234–244). PMLR.

Qendro, L., Chauhan, J., Ramos, A. G. C., & Mascolo, C. (2021). The benefit of the doubt: Uncertainty aware sensing for edge computing platforms. In *2021 IEEE/ACM symposium on edge computing* (pp. 214–227). IEEE.

Shi, M., Zhou, Y., Wang, K., Zhang, H., Huang, S., Ye, Q., & Lv, J. (2024). PRIOR: Personalized prior for reactivating the information overlooked in federated learning. *Advances in Neural Information Processing Systems*, *36*.

Shridhar, K., Laumann, F., & Liwicki, M. (2018). Uncertainty estimations by softplus normalization in bayesian convolutional neural networks with variational inference. arXiv preprint arXiv:1806.05978.

Shridhar, K., Laumann, F., & Liwicki, M. (2019). A comprehensive guide to bayesian convolutional neural network with variational inference. arXiv preprint arXiv:1901.02731.

Sun, M., Li, J., Guo, Z., Yu, Z., Zheng, Y., Si, X., & Liu, Z. (2016). Thuctc: an efficient chinese text classifier. GitHub Repository.

Triastcyn, A., & Faltings, B. (2019). Federated learning with bayesian differential privacy. In *2019 IEEE international conference on big data (big data)* (pp. 2587–2596). IEEE.

Wahab, O. A., Mourad, A., Otrok, H., & Taleb, T. (2021). Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems. *IEEE Communications Surveys & Tutorials*, *23*(2), 1342–1397.

Wang, J., Pal, A., Yang, Q., Kant, K., Zhu, K., & Guo, S. (2023). Collaborative machine learning: Schemes, robustness, and privacy. *IEEE Transactions on Neural Networks and Learning Systems*, *34*(12), 9625–9642. http://dx.doi.org/10.1109/TNNLS.2022.3169347.

Wang, Z., Zhu, Y., Wang, D., & Han, Z. (2021). Fedacs: Federated skewness analytics in heterogeneous decentralized data environments. In *2021 IEEE/ACM 29th international symposium on quality of service* (pp. 1–10). IEEE.

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

Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, *10*(2), 1–19.

Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, N., & Khazaeni, Y. (2019). Bayesian nonparametric federated learning of neural networks. In *International conference on machine learning* (pp. 7252–7261). PMLR.

Zhang, X., Li, Y., Li, W., Guo, K., & Shao, Y. (2022). Personalized federated learning via variational bayesian inference. In *International conference on machine learning* (pp. 26293–26310). PMLR.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., & Chandra, V. (2018). Federated learning with non-iid data. arXiv preprint arXiv:1806.00582.