DOI:
10.1109/LCOMM.2022.3223655

Document Version
Peer reviewed version

Compressed Particle-Based Federated Bayesian Learning and Unlearning

Jinu Gong, *Student Member, IEEE,* Osvaldo Simeone, *Fellow, IEEE,* and Joonhyuk Kang, *Member, IEEE*

***Abstract*—Conventional frequentist FL schemes are known to yield overconfident decisions. Bayesian FL addresses this issue by allowing agents to process and exchange uncertainty information encoded in distributions over the model parameters. However, this comes at the cost of a larger per-iteration communication overhead. This letter investigates whether Bayesian FL can still provide advantages in terms of calibration when constraining communication bandwidth. We present compressed particle-based Bayesian FL protocols for FL and federated "unlearning" that apply quantization and sparsification across multiple particles. The experimental results confirm that the benefits of Bayesian FL are robust to bandwidth constraints.**

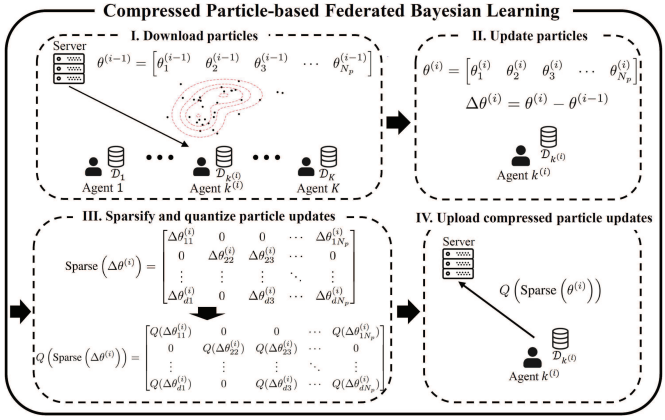
***Index Terms*—Federated learning, Bayesian learning, Stein variational gradient descent, Machine unlearning, Wireless communication**

I. Introduction

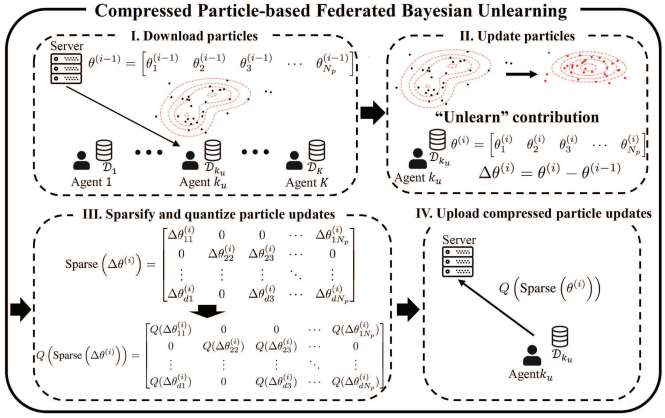
DISTRIBUTED intelligence is envisaged to be one of the key use cases for 6G. An important primitive for the implementation of distributed intelligence is federated learning (FL), which supports distributed gradient-based training across a network of learning agents (see [1] for an overview). Individual agents are often mobile devices with limited data and power [2], [3]. Despite such limitations, the decisions made by machine learning models trained via FL are expected to be used for sensitive applications such as personal healthcare. Furthermore, in such cases, agents may exercise their *right to be forgotten*, requesting that information about their data be "removed" from trained models available in the network for use by other devices [4]. This paper addresses the problem of developing communication-efficient FL protocols offering a reliable quantification of uncertainty while also supporting the right to erasure.

Most studies on FL are conducted within a frequentist framework, whereby agents perform local optimization in the space of model parameters, and iteratively exchange information about the updated model parameters through a server. Given the limited data available at each agent, there is uncertainty about the model parameters that are best suited to generalize outside the training set. By neglecting such uncertainty, frequentist learning schemes are known to yield poorly *calibrated* decisions, which are typically overconfident [5], [6]. Furthermore, in an FL framework, the "collapse" of uncertainty in the model parameter space – also known as epistemic uncertainty – to a single model parameter vector prevents agents from properly communicating their respective states of knowledge about the problem. This, in turn, can yield slower convergence [7].

A possible solution to this problem lies in adapting Bayesian learning methods, and generalizations thereof [1], [8], [9], to FL. Bayesian learning optimizes probability distributions over the model parameter space, allowing for a representation of the state of epistemic uncertainty caused by limited data at the agents. Practical implementations of Bayesian learning represent the model parameter distribution either via a parametric



(a) Compressed particle-based federated Bayesian learning



(b) Compressed particle-based federated Bayesian unlearning

Fig. 1: Compressed Particle-based federated Bayesian learning and unlearning.

The work of J. Gong and J. Kang was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2020-0-01787) supervised by the IITP (Institute of Information & Communications Technology Planning & Evaluation). The work of O. Simeone was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 725731)

Jinu Gong and Joonhyuk Kang are with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, 34141 Korea (e-mail: kjw7419@kaist.ac.kr, jhkang@kaist.edu).

Osvaldo Simeone is with the Department of Engineering, King's College London, London WC2R 2LS, U.K. (e-mail: osvaldo.simeone@kcl.ac.uk).

family of distributions – an approach known as variational inference (VI) – or via a set of random particles – following Monte Carlo (MC) sampling methods. MC-based methods can accurately estimate the target posterior distribution in the asymptotic regime of a large number of iterations, but they suffer from slow convergence. In contrast, VI-based methods have a significantly lower iteration complexity, but their performance is limited by the bias caused by the choice of a parametric family.

Stein variational gradient descent (SVGD) [10] is a non-parametric VI method that strikes a balance between expressivity of the approximation and iteration complexity. SVGD approximates the posterior distribution using a set of particles, like MC sampling, while also benefiting from the faster convergence of VI through deterministic optimization, rather than sampling. The *Distributed SVGD* (DSVGD) protocol introduced in [7] extends SVGD to FL (see Fig. 1a). The authors demonstrate the advantages of DSVGD in terms of the number of iterations and in terms of calibration with respect to standard frequentist FL. DSVGD was later adapted in [11] to introduce *Forget-DSVGD*, a protocol that accommodates the right to erasure by leveraging the VI framework for machine unlearning presented in [12] (see Fig. 1b).

Previous work [7] has assumed the possibility to transfer an unlimited amount of information at each iteration round. Therefore, the advantages highlighted in [7] of Bayesian FL were obtained at the cost of larger *per-iteration* communication overhead. In fact, in DSVGD, agents need to exchange multiple particles at each iteration, rather than a single model parameter vector as in frequentist FL. This paper investigates the question of whether Bayesian FL can still provide advantages in terms of iteration complexity and calibration when constraining communication bandwidth between agents and server. To address this problem, we present compressed DSVGD and Forget-DSVGD, which apply quantization and sparsification across multiple particles. While quantization and sparsification have been widely applied to frequentist FL [13], this is the first work to consider such techniques to reduce the communication overhead of particle-based Bayesian FL.

The rest of this letter is organized as follows. Sec. II introduces the setup, and reviews the necessary background. Sec. III presents the proposed compressed-DSVGD algorithm. Sec. IV describes numerical results, and Sec. V concludes the paper.

II. System Setup and Preliminaries

A. Setup

As illustrated in Fig. 1, we consider a federated learning setup with a set $\mathcal{K} = \{1, \ldots, K\}$ of K agents within a parameter-server architecture (see, e.g., [1]). The local data set $\mathcal{D}_k = \{z_{k,n}\}_{n=1}^{N_k}$ of agent $k \in \mathcal{K}$ contains N_k data points. The collection of all local datasets is referred to as the global data set $\mathcal{D}$. We express the local training loss at agent k with respect to the $d \times 1$ model parameter θ as the empirical average

$$L_k(\theta) = \frac{1}{N_k} \sum_{n=1}^{N_k} \ell(z_{k,n}|\theta) \tag{1}$$

for some loss function $\ell(z|\theta)$. For a likelihood function $p(z|\theta)$, the loss function is typically chosen as the log-loss $\ell(z|\theta) = -\log p(z|\theta)$.

In Bayesian federated learning, the goal is to obtain a *variational distribution* $q(\theta)$ on the model parameter space that minimizes the *global free energy* (see, e.g., [1], [7], [14])

$$\min_{q(\theta)} \left\{ F(q(\theta)) = \sum_{k=1}^{K} N_k \mathbb{E}_{\theta \sim q(\theta)}[L_k(\theta)] + \alpha \cdot \mathbb{D}\big(q(\theta) \big\| p_0(\theta)\big) \right\}, \tag{2}$$

where $\alpha > 0$ is a "temperature" parameter; $\mathbb{D}(\cdot\|\cdot)$ is the Kullback–Leibler (KL) divergence; and $p_0(\theta)$ denotes a prior distribution. The optimization problem (2) seeks for a distribution $q(\theta)$ that minimizes the average sum-training loss, i.e., the first term in (2), while being close to the prior distribution $p_0(\theta)$, as enforced by the second term.

The unconstrained optimal solution of problem (2) is given by the *global generalized posterior distribution*

$$q^*(\theta|\mathcal{D}) = \frac{1}{Z} \cdot \tilde{q}^*(\theta|\mathcal{D}) \tag{3}$$

$$\text{where} \quad \tilde{q}^*(\theta|\mathcal{D}) = p_0(\theta) \exp\left(-\frac{1}{\alpha} \sum_{k=1}^{K} N_k L_k(\theta) \right), \tag{4}$$

which equals the conventional posterior distribution $p\big(\theta|\mathcal{D}\big)$ when one sets $\alpha = 1$ and the loss function as the *log-loss* $\ell(z|\theta) = -\log p(z|\theta)$.

However, in practice, problem (2) can only be solved in an approximate manner by using parametric or non-parametric methods. In this letter, we focus on a state-of-the-art non-parametric particle-based method, SVGD [10], which represents the distribution $q(\theta)$ in (2) in terms of N_p particles $\{\theta_1, \ldots, \theta_{N_p}\}$ (see Fig. 1). Given particles $\{\theta_1, \ldots, \theta_{N_p}\}$, an explicit estimate of distribution $q(\theta)$ can be obtained, e.g., via kernel density estimator (KDE) with some kernel function $K(\theta, \theta')$, i.e., $q(\theta) = \frac{1}{N_p} \sum_{n=1}^{N_p} K(\theta, \theta_n)$ (see, e.g., [1]).

B. Distributed SVGD

DSVGD addresses problem (2) in a federated setting by describing distribution $q(\theta)$ via a set of N_p particles $\{\theta_n\}_{n=1}^{N_p}$ that are updated by scheduling a subset of agents each iteration (see, e.g., [7]). In this letter, we focus on the case of a single agent scheduled at each iteration, since the extension to more than one agent is direct by following the approach in [7].

At the beginning of the i-th iteration, the server stores the current global particles $\{\theta_n^{(i-1)}\}_{n=1}^{N_p}$, which represent the current iterate $q^{(i-1)}(\theta)$ of the global variational distribution. The variational distribution $q^{(i-1)}(\theta)$ is modelled via the factorization $q^{(i-1)}(\theta) = p_0(\theta) \prod_{k=1}^{K} t_k^{(i-1)}(\theta)$ [1], [7], [14], where the term $t_k^{(i-1)}(\theta)$ is known as approximate likelihood of agent k. At each iteration i, the scheduled agent $k^{(i)}$ updates the variational distribution $q^{(i-1)}(\theta)$ by modifying its approximate likelihood to a new iterate $t_{k^{(i)}}^{(i)}(\theta)$ via the optimization of a set of local particles. Specifically, given kernel functions $K(\cdot,\cdot)$ and $\kappa(\cdot,\cdot)$, DSVGD operate as follows [7].

Initialization. Draw the set of N_p global particles $\{\theta_n^{(0)}\}_{n=1}^{N_p}$ from the prior $p_0(\theta)$; and initialize at random the set of local particles $\{\theta_{k,n}^{(0)}\}_{n=1}^{N_p}$ for all agents $k \in \mathcal{K}$.

Step 1. At each iteration i, server schedules an agent $k^{(i)} \in \mathcal{K}$. Agent $k^{(i)}$ downloads the current global particles $\{\theta_n^{(i-1)}\}_{n=1}^{N_p}$ from the server.

Step 2. Agent $k^{(i)}$ initializes its particles to equal the global particles, i.e., $\{\theta_n^{[0]} = \theta_n^{(i-1)}\}_{n=1}^{N_p}$. Furthermore, it sets its local likelihood to $t_{k^{(i)}}^{(i-1)}(\theta) = 1/N_p \sum_{n=1}^{N_p} K(\theta, \theta_{k^{(i)},n}^{(i-1)})$ and the global posterior to $q^{(i-1)}(\theta) = 1/N_p \sum_{n=1}^{N_p} K(\theta, \theta_n^{(i-1)})$. Then, it updates the particles via SVGD [7] as

$$\theta_n^{[l]} \leftarrow \theta_n^{[l-1]} + \epsilon \phi\left(\theta_n^{[l-1]}\right), \tag{5}$$

for all particles $n = 1, \ldots, N_p$ with learning rate ϵ, and function

$$\phi(\theta) = \frac{1}{N_p} \sum_{j=1}^{N_p} \Bigg[\kappa\left(\theta_j^{[l-1]}, \theta\right) \nabla_{\theta_j} \log \tilde{p}_{k^{(i)}}^{(i)}\left(\theta_j^{[l-1]}\right) + \nabla_{\theta_j} \kappa\left(\theta_j^{[l-1]}, \theta\right) \Bigg], \tag{6}$$

across local iterations $l = 1, \ldots, L$, where we have defined the "tilted" distribution as

$$\tilde{p}_{k^{(i)}}^{(i)}(\theta) \propto \frac{q^{(i-1)}(\theta)}{t_{k^{(i)}}^{(i-1)}(\theta)} \exp\left(-\frac{1}{\alpha} L_{k^{(i)}}(\theta)\right). \tag{7}$$

Step 3. After L local iteration, agent $k^{(i)}$ sets $\{\theta_n^{(i)} = \theta_n^{[L]}\}_{n=1}^{N_p}$. The updated global particles $\{\theta_n^{(i)}\}_{n=1}^{N_p}$ are sent to the server, which sets $\{\theta_n = \theta_n^{(i)}\}_{n=1}^{N_p}$. Finally, agent $k^{(i)}$ updates its local particles $\{\theta_{k^{(i)},n}^{(i)}\}_{n=1}^{N_p}$ using the updated global particles $\{\theta_n^{(i)}\}_{n=1}^{N_p}$, while the other agents $k' \neq k^{(i)}$ set $\{\theta_{k',n}^{(i)} = \theta_{k',n}^{(i-1)}\}_{n=1}^{N_p}$. We refer to [7, Sec. 5.2] for benefits on the update of the local particles.

C. Forget-SVGD

We finally describe the variational unlearning formulation in [12], which is referred to as Forget-SVGD. Before unlearning, Forget-SVGD assumes that an approximate solution $q(\theta|\mathcal{D})$ of the federated learning problem (2) has been obtained, e.g., via DSVGD. Forget-SVGD aims at removing the contribution for data of a subset $\mathcal{U} \subset \mathcal{K}$ of agents, which wish to unlearn, from the learned model $q(\theta|\mathcal{D})$.

A baseline approach would retrain *from scratch* the global model excluding the agents in subset $\mathcal{U}$. A potentially more efficient solution, Forget-SVGD, operates as follows [11].

Initialization. The initial set of N_p particles $\{\theta_n^{(0)}\}_{n=1}^{N_p}$ represents the variational distribution obtained as a result of Bayesian federated learning; initialize at random local particles $\{\theta_{k,n}^{(0)}\}_{n=1}^{N_p}$ for all agents $k \in \mathcal{U}$

Step 1. At iteration i, the server schedules an agent $k^{(i)} \in \mathcal{U}$, within the set of agents who have requested their data to be "forgotten". Agent $k^{(i)}$ downloads the current global particles $\{\theta_n^{(i-1)}\}_{n=1}^{N_p}$ from the server.

Step 2. Agent $k^{(i)}$ initializes the particles $\{\theta_n^{[0]} = \theta_n^{(i-1)}\}_{n=1}^{N_p}$, and it updates the particles using the SVGD update (5)-(6) by replacing the tilted distribution in (7) with

$$\tilde{p}_{k^{(i)}}^{(i)}(\theta) = \frac{q^{(i-1)}(\theta)}{t_{k^{(i)}}^{(i-1)}(\theta)} \exp\left(\frac{1}{\alpha} L_{k^{(i)}}(\theta)\right), \tag{8}$$

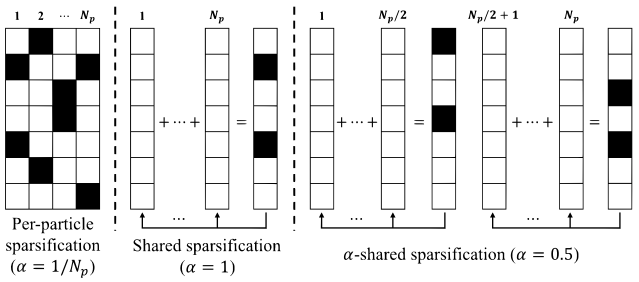


Fig. 2: Sparsification methods.

where $q^{(i-1)}(\theta)$ and $t_{k^{(i)}}^{(i-1)}(\theta)$ are computed by using the respective KDEs with global and local particles, respectively.

Step 3. This step applies the same operations as Step 3 of DSVGD.

III. Compressed-Distributed Stein Variational Gradient Descent

In this section, we study an implementation of DSVGD and Forget-SVGD over rate-constrained channels between agents and server. This constraint affects the uploading of the updated global particles $\{\theta_n^{(i)}\}_{n=1}^{N_p}$ at each i-th iteration of Step 3 of DSVGD and Forget-SVGD. Downlink communication from server to agents is assumed to be noiseless in order to focus on the more challenging uplink channel. Accordingly, the scheduled agent $k^{(i)}$ can communicate no more than R_u bits per iteration.

To facilitate compression, at each i-th iteration, agent $k^{(i)}$ uploads the $N_p \times d$ matrix $\Delta\Theta^{(i)} = \left[\theta_1^{(i)} - \theta_1^{(i-1)}, \theta_2^{(i)} - \theta_2^{(i-1)}, \ldots, \theta_{N_p}^{(i)} - \theta_{N_p}^{(i-1)}\right]$ of updates for all particles to the server. This is a common step in compressed frequentist FL algorithms, which communicate a single parameter vector per iteration, i.e., they have $N_p = 1$ [13]. In this section, we develop compression strategies based on sparsification and quantization of the updates $\Delta\Theta^{(i)}$ in order to meet the capacity constraint of R_u bits per iteration. The key novel element as compared to prior work is the need to compress multiple particles simultaneously.

A. Sparsification

Top-k sparsification is a widely used method for frequentist FL that selects the entries of the model parameter vector with the k largest absolute values. All other entries of the update vector are set to zero. In this work, we introduce and study the following variants of top-k sparsification for particle-based Bayesian learning based on DSVGD. For all schemes, to identify the sparsified position, we assume Golomb position encoding, which requires $\log_2 \binom{d}{k}$ for an input vector of d entries [15]. We denote as N_b the number of bits used to represent each entry retained by the sparsification process.

1) Per-particle sparsification: A baseline approach is to apply top-k sparsification separately to each particle update (see Fig. 2-(left)). This scheme requires

$$R_u = N_p \times \left(\log_2 \binom{d}{r \times d} + N_b \times r \times d\right) \tag{9}$$

bits per iteration, where we have defined the per-particle sparsification ratio $r = k/d$, and the first term in the parenthesis

is the number of bits required to specify the top-k positions in each particle. The communication overhead necessary to identify the top-k entries hence scales linearly with the number N_p of particles.

2) Shared sparsification: When the bit rate R_u is small, a potentially more efficient approach is based on the assumption that the sparsity pattern is common to all particles. To implement this idea, which we refer to as shared sparsification, we sum the absolute values of each entry of the N_p particles, and the top-k entries are selected based on the resulting sum vector. The resulting sparsity pattern is applied to all particles (see Fig. 2-(center)). This scheme requires

$$R_u = \log_2 \binom{d}{r \times d} + N_p \times N_b \times r \times d \tag{10}$$

bits per iteration, reducing by N_p times the overhead for position encoding.

3) α-shared sparsification: Generalizing the previous two schemes, α-shared sparsification divides the particles into $1/\alpha$ groups, and only shares the sparsity pattern among particles in the same group. For each group, the scheme applies the same procedure of shared sparsification method (see Fig. 2-(right)). Note that setting $\alpha = 1/N_p$ yields per-particle sparsification; and setting $\alpha = 1$ yields shared sparsity. More generally, the scheme is defined for every value $\alpha \in [1/N_p, 1]$ such that $1/\alpha$ is an integer that divides N_p. This scheme requires

$$R_u = \frac{1}{\alpha} \times \log_2 \binom{d}{r \times d} + N_p \times N_b \times r \times d \tag{11}$$

bits per iteration, reducing by N_p times the overhead for position encoding.

B. Quantization

Every entry selected by the sparsification step is finally quantized using stochastic quantization [13]. For each entry $x \in \mathbb{R}$, the scheme requires 1 bit for the sign $\text{sign}(x)$, and $N_b - 1$ bits for the magnitude $|x|$. Within a predefined dynamic range $[0, a_{\max}]$, a step size $\delta = a_{\max}/(2^{N_b} - 1)$ is set, and the stochastic quantizer $Q_{N_b}(x)$ is defined as

$$Q_{N_b}(x) = \text{sign}(x) \cdot \zeta(\text{clip}(|x|), N_b), \tag{12}$$

where

$$\zeta(a, N_b) = \begin{cases} t\delta & \text{with probability } 1 - \frac{a - t\delta}{\delta} \\ (t+1)\delta & \text{otherwise,} \end{cases} \tag{13}$$

and $\text{clip}(a) = \min(a, a_{\max})$.

IV. Experiments

A. Federated Learning

We are interested in comparing the performance of frequentist FL and Bayesian FL in the presence of an uplink per-iteration rate constraint R_u. For frequentist FL, we adopt FedAvg with standard top-k sparsification and stochastic quantization as in, e.g., [13], [16]. We have $K = 10$ agents, each with $N_k = 6000$ examples from the Fashion-MNIST data set. The model consists of one fully-connected hidden layer with 100 hidden neurons and a softmax output layer. For compressed-DSVGD, as in [10], we consider the radial basis function (RBF) kernel $\kappa(x, x') = \exp(-\|x - x'\|_2^2/h)$

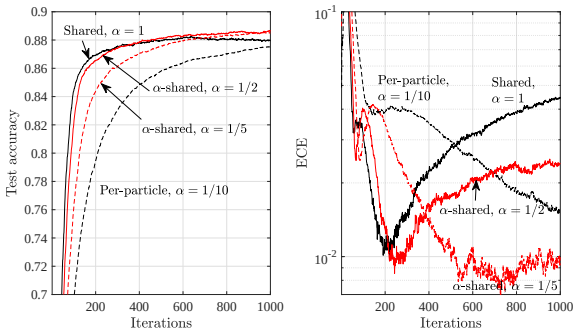


Fig. 3: Test accuracy and ECE for three sparsification methods with $R_u = d$ bits per iteration (Fashion-MNIST data set, $K = 10$ devices).

and the bandwidth $h = \text{med}^2/\log N$, where med is the median of the pairwise distances between the particles in the current iteration. We also assume the Gaussian kernel $K(x, x') \propto \exp(-\|x - x'\|^2/\lambda)$ for the KDE with a bandwidth $\lambda = 0.55$. The fixed temperature parameter is set to $\alpha = 1$, and AdaGrad [10] is used to determine the learning rate schedule in (5).

We evaluate the performance by using two metrics, namely test accuracy and *expected calibration error* (ECE) [5]. The ECE measures the capacity of a model to quantify uncertainty. It does so by evaluating the difference between the confidence level output by the model and the actual test accuracy. The confidence level is given by the output of the last, softmax, layer corresponding to the prediction of the model. The ECE is defined by partitioning the test set into M bins $\{B_m\}_{m=1}^M$ depending on the confidence level of the model's decision, and by evaluating the accuracy $\text{acc}(B_m)$ for the examples within each bin. The ECE is given by the average of the difference between accuracy $\text{acc}(B_m)$ and confidence $\text{conf}(B_m)$ across all bins as [5]

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left|\text{acc}(B_m) - \text{conf}(B_m)\right|, \tag{14}$$

where $|B_m|$ is the number of test examples in the m-th bin.

We start by comparing the performance of compressed-DSVGD under the proposed sparsification methods by setting the number of particle to $N_p = 10$, the number of quantization bits to $N_b = 5$, and the per-iteration rate to $R_u = d$. Fig. 3 plots test accuracy and ECE as a function of the training iterations, where average results are reported over 10^2 runs of the algorithms. The figure suggests that, while shared sparsification is most effective when we can only run a small number of iterations, shared sparsification with $\alpha < 1$ is required to obtain smaller values of test error and ECE. Note that the minimum value of α, $\alpha = 1/10$, which corresponds to per-particle sparsification is generally suboptimal.

In Fig. 4, we present test accuracy with respect to ECE after 10^3 training iterations for FedAvg and DSVGD with $N_p = 2, 5, 10$ particles. We apply α-shared sparsity and vary the per-iteration bits constraints $R_u = 0.5d, d, 5d, 10d$. As shown in Fig. 4, DSVGD outperforms FedAvg in test accuracy and ECE, even under the per-iteration bit constraints, unless the number

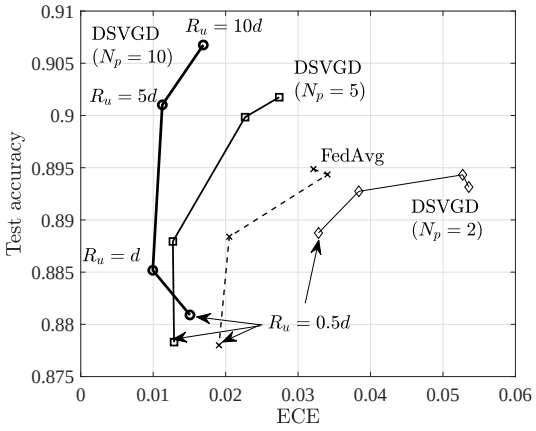


Fig. 4: Test accuracy and ECE plot for FedAvg and DSVGD with $N_p = 2, 5, 10$ particles under per-iteration bit constraints $R_u = 0.5d, d, 5d, 10d$. The markers indicate points with the same value of R_u.

of particles, N_p, is too low, here $N_p = 2$. Furthermore, as a per-iteration bits constraint R_u decreases, it is more beneficial to reduce the number of particles N_p than to decrease the sparsification ratio r.

B. Federated Unlearning

For federated unlearning, we adopt a "non-iid" setting with $K = 10$ agents by assigning each agent 100 examples from only two of the ten classes of Fashion-MNIST images. The two agents with labels 2 and 9 request that their contribution be "unlearned". We follow references [17], [18] by pre-training using conventional FedAvg, and then training the last layer using DSVGD with $N_p = 40$ particles. Then, we "unlearn" the model based on the proposed compressed Forget-SVGD scheme. Fig. 5-(left) shows the average test accuracy for the unlearned labels (2 and 9) and that of remaining labels during compressed-Forget-SVGD iterations for per-iteration bit constraints $R_u = d, 0.5d$. The right panel shows, for reference, the performance of a train-from-scratch scheme using only the remaining labels, which is seen to be significantly slower. For a smaller bandwidth R_u, here $R_u = 0.5d$, using a larger α tends to degrade, as desired, the accuracy for the unlearned labels, while also affecting the performance of the other labels. This points to a trade-off between forgetting and retraining useful information that can be controlled via the parameter α.

V. Conclusion

This letter has investigated the performance of particle-based Bayesian federated learning and unlearning under bandwidth constraints. A new class of sparsification methods was proposed that operates across multiple particles. Through simulations, we have confirmed that Bayesian FL can outperform standard frequentist FL in terms of test accuracy and calibration even under per-iteration bit constraints. Furthermore, we have identified a trade-off between forgetting requested data and retraining useful information that can be controlled by the choice of the sparsification scheme.

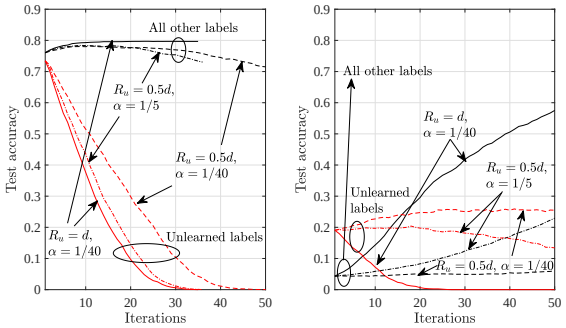


Fig. 5: Average test accuracy of the unlearned labels and of the remaining labels for unlearning compressed-Forget-SVGD and the training from scratch using the data set $\mathcal{D} \setminus \mathcal{D}_u$ via compressed-DSVGD under per bit constraints $R_u = d$, $0.5d$.

References

[1] O. Simeone, *Machine Learning for Engineers*. Cambridge University Press, 2022.

[2] I. Kholod *et al.*, "Open-source federated learning frameworks for IoT: A comparative review and analysis," *Sensors*, vol. 21, no. 1, p. 167, Dec. 2020.

[3] C.-R. Shyu *et al.*, "A systematic review of federated learning in the healthcare area: From the perspective of data properties and applications," *Applied Sciences*, vol. 11, no. 23, p. 11191, Nov. 2021.

[4] A. Ginart, M. Guan, G. Valiant, and J. Y. Zou, "Making AI forget you: Data deletion in machine learning," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[5] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International Conference on Machine Learning*, 2017, pp. 1321–1330.

[6] M. E. Khan and H. Rue, "The Bayesian learning rule," *arXiv preprint arXiv:2107.04562*, 2021.

[7] R. Kassab and O. Simeone, "Federated generalized Bayesian learning via distributed stein variational gradient descent," *IEEE Trans. Sig. Proc.*, vol. 70, pp. 2180–2192, 2022.

[8] J. Knoblauch, J. Jewson, and T. Damoulas, "Generalized variational inference: Three arguments for deriving new posteriors," *arXiv preprint arXiv:1904.02063*, 2019.

[9] S. T. Jose and O. Simeone, "Free energy minimization: A unified framework for modeling, inference, learning, and optimization [lecture notes]," *IEEE Sig. Proc. Mag.*, vol. 38, no. 2, pp. 120–125, Mar. 2021.

[10] Q. Liu and D. Wang, "Stein variational gradient descent: A general purpose Bayesian inference algorithm," in *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[11] J. Gong, J. Kang, O. Simeone, and R. Kassab, "Forget-SVGD: Particle-Based Bayesian Federated Unlearning," in *IEEE Data Science and Learning Workshop*, 2022.

[12] Q. P. Nguyen, B. Kian, H. Low, and P. Jaillet, "Variational Bayesian unlearning," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020, pp. 16 025–16 036.

[13] D. Alistarh *et al.*, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Advances in neural information processing systems*, vol. 30, 2017, pp. 1707–1718.

[14] T. D. Bui, C. V. Nguyen, S. Swaroop, and R. E. Turner, "Partitioned variational inference: A unified framework encompassing federated and continual learning," *arXiv preprint arXiv:1811.11206*, 2018.

[15] F. Sattler *et al.*, "Sparse binary compression: Towards distributed deep learning with minimal communication," in *International Joint Conference on Neural Networks*, 2019, pp. 1–8.

[16] Z. Qin *et al.*, "Federated learning and wireless communications," *IEEE Wireless Commun.*, vol. 28, no. 5, pp. 134–140, Oct. 2021.

[17] A. Kristiadi, M. Hein, and P. Hennig, "Being Bayesian, even just a bit, fixes overconfidence in ReLu networks," in *International Conference on Machine Learning*, 2020, pp. 5436–5446.

[18] L. V. Jospin, H. Laga, F. Boussaid, W. Buntine, and M. Bennamoun, "Hands-on Bayesian neural networks—A tutorial for deep learning users," *IEEE Comput. Intell. Mag.*, vol. 17, no. 2, pp. 29–48, May 2022.