Compressed Particle-Based Federated Bayesian Learning and Unlearning

Jinu Gong\textsuperscript{\textcopyright}, Student Member, IEEE, Osvaldo Simeone\textsuperscript{\textcopyright}, Fellow, IEEE, and Joonhyuk Kang\textsuperscript{\textcopyright}, Member, IEEE

Abstract—Conventional frequentist federated learning (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 letter addresses the problem of developing communication-efficient FL protocols that offer 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 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. Applications of Bayesian learning to communication systems include [10], [11].

Stein variational gradient descent (SVGD) [12] 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 [13] to introduce Forget-DSVGD, a protocol that accommodates the right to erasure by leveraging the VI framework for machine unlearning presented in [14] (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 letter 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...
Compressed Particle-based Federated Bayesian Learning

I. Download particles

\[ \{\theta^{(n)}\}_{n=1}^{N_k} \]

II. Update particles

\[ \Delta \theta^{(t)} = \theta^{(t)} - \theta^{(t-1)} \]

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

(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], [16])
\[
\min_{q(\theta)} \left\{ F(q(\theta)) = \sum_{k=1}^{K} N_k \mathbb{E}_{q(\theta)}[L_k(\theta)] + \alpha \cdot \mathbb{D}(q(\theta)||p_0(\theta)) \right\},
\]

(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 distribu-
tion \( q(\theta) \) 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
\[
q^*(\theta|D) = \frac{1}{Z} \cdot \tilde{q}^*(\theta|D)
\]

(3)

where \( \tilde{q}^*(\theta|D) = p_0(\theta) \exp \left( -\frac{1}{\alpha} \sum_{k=1}^{K} N_k L_k(\theta) \right) \)

(4)

which equals the conventional posterior distribution \( p(\theta|D) \)
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 [12], which re-
distributes 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

DSVG 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 \( \{\theta^{(i-1)}_n\}_{n=1}^{N_p} \), which represent
the current iterate \( \tilde{q}^{(i-1)}(\theta) \) of the global variational distribution.
The variational distribution \( q^{(i-1)}(\theta) \) is modelled via the
factorization \( q^{(i-1)}(\theta) = p_0(\theta) \prod_{k=1}^{K} \tilde{q}^{(i-1)}_k(\theta) \) [1], [7], [16],
where the term \( \tilde{q}^{(i-1)}_k(\theta) \) 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 approxi-
mate likelihood to a new iterate \( \tilde{q}^{(i)}_k(\theta) \) via the optimization
of a set of local particles. Specifically, given kernel functions
\( K(\cdot, \cdot) \) and \( \kappa(\cdot, \cdot) \), DSVG operate as follows [7].

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

Step 1: At each iteration \( i \), server schedules an agent
\( k^{(i)} \in K \). Agent \( k^{(i)} \) downloads the current global particles
\( \{\theta^{(i-1)}_n\}_{n=1}^{N_p} \) 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 $t_k(i-1) = 1/N_p \sum_{n=1}^{N_p} K(\theta, \theta_n^{(i-1)})$ 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
\[
\theta_n^{(l+1)} = \theta_n^{(l)} + \epsilon \phi \left( \frac{\theta_n^{(l)}}{\theta_n^{(l)}} \right),
\]
for all particles $n = 1, \ldots, N_p$, with learning rate $\epsilon$, and function
\[
\phi(\theta) = \frac{1}{N_p} \sum_{j=1}^{N_p} \kappa \left( \theta_j^{(l)} - \theta_j^{(l-1)} \right) \nabla_{\theta_j} \log p(\theta_j^{(l)}) + \nabla_{\theta_j} \kappa \left( \theta_j^{(l-1)} - \theta_j^{(l-1)} \right),
\]
across local iterations $l = 1, \ldots, L$, where we have defined the “tilted” distribution as
\[
\tilde{p}(\theta) = \frac{q(i-1)(\theta)}{q(i-1)(\theta)} \exp \left( -\frac{1}{\alpha} L_k(i)(\theta) \right).
\]

Step 3: After $L$ local iteration, agent $k(i)$ sets $\{\theta_n^{(L)} \}_{n=1}^{N_p}$. The updated global particles $\{\theta_n^{(L)} \}_{n=1}^{N_p}$ are sent to the server, which sets $\{\theta_n = \theta_n^{(L)} \}_{n=1}^{N_p}$. Finally, agent $k(i)$ updates its local particles $\{\theta_n^{(k)} \}_{n=1}^{N_p}$ using the updated global particles $\{\theta_n^{(L)} \}_{n=1}^{N_p}$, while the other agents $k' \neq k(i)$ set $\{\theta_n^{(k')} = \theta_n^{(L)} \}_{n=1}^{N_p}$. 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 [14], which is referred to as Forget-SVGD. Before unlearning, Forget-SVGD assumes that an approximate solution $q(\theta|D)$ 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 $U \subset K$ of agents, which wish to unlearn, from the learned model $q(\theta|D)$.

A baseline approach would retrain from scratch the global model excluding the agents in subset $U$. A potentially more efficient solution, Forget-SVGD, operates as follows [13].

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 $\{\theta_n^{(0)} \}_{n=1}^{N_p}$ for all agents $k \in U$.

Step 1: At iteration $i$, the server schedules an agent $k(i)$ to download 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
\[
\tilde{p}(\theta) = \frac{q(i-1)(\theta)}{q(i-1)(\theta)} \exp \left( -\frac{1}{\alpha} L_k(i)(\theta) \right),
\]
where $q(i-1)(\theta)$ and $q(i-1)(\theta)$ are computed by using the respective KDEs with global and local particles, respectively.

Step 3: The same operations are applied as Step 3 of DSVGD.
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

\[
R_u = \log_2 \left( \frac{d}{r \times d} \right) + N_p \times N_b \times r \times d
\]

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

3) \( \alpha \)-Shared Sparsification: Generalizing the previous two schemes, \( \alpha \)-shared sparsification divides the particles into \( \lfloor 1/\alpha \rfloor \) groups, and only shares the sparsity pattern among particles in the same group. For each group, the scheme applies the same procedure of the 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

\[
R_u = \frac{1}{\alpha} \times \log_2 \left( \frac{d}{r \times d} \right) + N_p \times N_b \times r \times d
\]

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 [15]. 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 \([0, a_{\text{max}}]\), a step size \( \delta = a_{\text{max}}/(2^{N_b} - 1) \) is set, and the stochastic quantizer \( Q_{N_b}(x) \) is defined as

\[
Q_{N_b}(x) = \text{sign}(x) \cdot \xi(x) = \begin{cases} 
\delta & \text{with probability } 1 - \frac{a_{\text{max}} - t\delta}{a_{\text{max}}} \\
(t+1)\delta & \text{otherwise}
\end{cases}
\]

where

\[
\xi(a) = \begin{cases} t\delta & \text{with probability } 1 - \frac{a_{\text{max}} - t\delta}{a_{\text{max}}} \\
(t+1)\delta & \text{otherwise}
\end{cases}
\]

\[
\text{clip}(a) = \min(a, a_{\text{max}})
\]

IV. EXPERIMENTS

A. Federated Learning

We are interested in comparing the performance of frequentist FL and Bayesian FL in the presence of an unpleasant per-iteration rate constraint \( R_u \). For frequentist FL, we adopt FedAvg with standard top-\( k \) sparsification and stochastic quantization as in, e.g., [15], [18]. 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 [12], we consider the radial basis function (RBF) kernel \( k(x, x') = \exp(-\|x - x'\|^2/h^2) \) and the bandwidth \( h = \text{med}^2/\log N \), where med is the median of the pairwise distances between the particles. 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 [12] 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]

\[
\text{ECE} = \frac{1}{M} \sum_{m=1}^M |B_m| \frac{|\text{acc}(B_m) - \text{conf}(B_m)|}{n}, \quad (13)
\]

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, 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 \( \alpha \), \( \alpha = 1/10 \), which corresponds to per-particle sparsification is generally suboptimal. We also observe that a larger number of iterations can cause overfitting in terms of likelihood, which in turn yields an increasing ECE [5, Fig. 3].

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. The number of iterations can be practically determined via validation by using plots like Fig. 3. We apply \( \alpha \)-shared sparsity and vary the per-iteration bits constraints \( R_u = 0.5d, 0.5d, 0.5d, 10d \). As shown in Fig. 4, DSVGD outperforms FedAvg in test accuracy and ECE, even under the per-iteration bit constraints, unless the number of particles, \( N_p \), is too low, here \( N_p = 2 \). Furthermore, as the per-iteration bits constraint \( R_u \) decreases, e.g., for \( R_u = 0.5d \), it may
be preferable to decrease the number of particles, e.g., from $N_p = 10$ to $N_p = 5$, when calibration is a more critical requirement than accuracy.

**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 in [19] and [20] 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 $\alpha$ 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 $\alpha$.

**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.