OPT2024: 16th Annual Workshop on Optimization for Machine Learning

Simple and Scalable Federated Learning with Uncertainty via Improved Variational Online Newton

Shivam Pal PSHIVAM@CSE.IITK.AC.IN
Aishwarya Gupta AISHWARYAG@CSE.IITK.AC.IN
Saqib Sarwar SAQIB@CSE.IITK.AC.IN
Piyush Rai PIYUSH@CSE.IITK.AC.IN
Department of Computer Science and Engineering IIT Kanpur, India

Abstract

The standard setup for federated learning (FL) is as a distributed optimization problem where each client learns a local model using its own private data, and these local models are aggregated at a central server. Recent works have also focused on settings where, not just the predictive accuracy but the model and predictive *uncertainty* estimates are also of interest. Bayesian FL methods have recently emerged as a promising way to achieve this by solving a distributed *posterior inference* problem. However, computing as well as aggregating the local client posteriors is usually much more expensive (both in terms of local computation as well as the client-server communication) than optimization based FL approaches, such as FedAvg. We present a simple and scalable Bayesian FL method in which, in each round, each client approximates its local posterior using the improved variational online Newton method, which has almost the same cost as simply running an Adam optimizer, making distributed inference mimic distributed optimization. We also present an efficient aggregation method for the client posteriors to learn the global model at the server. Our method achieves improved predictive accuracies as well as better uncertainty estimates as compared to the baselines which include both optimization based FL as well as Bayesian FL methods.

1. Introduction

Federated learning (FL) [18] aims to learn a combined model from different clients with the constraint that sharing data among clients is not allowed. However, weights learned at each client can be shared on the server, and at the server, these weights can be aggregated to learn the combined model. FedAvg [18], a popular FL algorithm, does a simple weight averaging at the server, which however is not ideal if there is data heterogeneity across clients. Moreover, these methods do not perform well when clients have very limited amounts of training data. In such settings, learning the posterior *distribution* at each client is more useful, as demonstrated in several recent works, such as [2, 4, 10, 16] which have advocated taking a Bayesian approach to FL. However, existing Bayesian FL methods usually rely on running computationally expensive routines on the clients (e.g., requiring expensive MCMC sampling [2], expensive Laplace's approximation which requires Hessian computations [16] on the clients, or methods based on learning deep ensembles [15]), as well as expensive client-server communication [11] and aggregation at the server (note that, unlike standard FL, Bayesian FL would require sending the whole client posterior to the server). Therefore, computationally efficient Bayesian FL methods are highly desirable.

In this work, we develop a Bayesian FL method which has an almost similar computational cost as standard FL methods such as FedAvg, that use fast optimizers such as SGD or Adam, while still providing the various benefits of a Bayesian approach such as robust predictions and quantification

of model and predictive uncertainty. In particular, we leverage the IVON (Improved Variational Online Newton) algorithm [20] to perform highly efficient variational inference (VI) on each client by approximating its local posterior using a Gaussian with diagonal covariance. Our method is computationally cheaper than other existing Bayesian FL methods that use expensive MCMC sampling [2, 4], Laplace's approximation [16], or even VI [11] at the clients. These local posteriors can be efficiently sent to the server and the global posterior can be computed for which we also present local posterior aggregation strategies.

2. Bayesian FL via Improved Variational Online Newton

The standard formulation of FL is similar to distributed optimization except some additional constraints, such as no data sharing among clients and server and a limited communication budget. Assuming K clients, let $\mathcal{D} = \bigcup_{k\in[K]} \mathcal{D}_k$ be the total available data where $\mathcal{D}_k$ denotes the private data of client k. The objective of standard FL is solve $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{k\in[K]} -\log p(\mathcal{D}_k \mid \boldsymbol{\theta})$. However, this optimization problem is not trivial as it requires access to each client's data which is not permitted in the federated setting. Thus, a multi-round approach is usually taken where clients learn their local models, send these local models to a central server which aggregates them into a global model, and send the global model to the clients to continue the next round of learning.

Unlike standard FL which only learns a point estimate of $\boldsymbol{\theta}$, an alternative is to learn a *distribution* of $\boldsymbol{\theta}$. The posterior distribution of $\boldsymbol{\theta}$ can be written as

$$p(\boldsymbol{\theta} \mid \mathcal{D}) \propto p(\boldsymbol{\theta}) \prod_{k\in[K]} p(\mathcal{D}_k \mid \boldsymbol{\theta}) \tag{1}$$

where $p(\boldsymbol{\theta})$ is prior distribution on $\boldsymbol{\theta}$ and $p(\mathcal{D}_k \mid \boldsymbol{\theta})$ is data likelihood of client k. Assuming uniform prior $p(\boldsymbol{\theta})$, it can be trivially shown that the optimizing the standard FL objective function is equivalent to finding the mode of the posterior $p(\boldsymbol{\theta} \mid \mathcal{D})$, i.e., $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta} \mid \mathcal{D})$.

Computing the full posterior $p(\boldsymbol{\theta} \mid \mathcal{D})$ is more useful than computing just the point estimate $\boldsymbol{\theta}^*$ because the posterior helps take into account model uncertainty. However, it is computationally intractable to compute the posterior exactly. Directly approximating $p(\boldsymbol{\theta} \mid \mathcal{D})$ using approximate inference methods such as MCMC or variational inference [3] is also non-trivial, as it requires computing each client's likelihood which in turn requires global access to all the client's data.

Claim 1 *The global posterior $p(\boldsymbol{\theta} \mid \mathcal{D})$ can be approximated at the server by the product of local client posteriors without requiring access to any client's local data.*

If local posteriors $p(\theta \mid \mathcal{D}_k)$ are also being approximated, multiple rounds of optimization are needed to reduce the aggregation error in the global posterior [2]. In FL, another challenge is to make the computation of the local posteriors, their aggregation at the server, and the client-server communication, efficient, which in general can be difficult even for simple models [2].

2.1. Client's posterior approximation

Assuming client k has N_k training examples, its local loss can be defined as $\bar{\ell}_k(\boldsymbol{\theta}) = \frac{1}{N_k} \sum_{i=1}^{N_k} \ell_i(\boldsymbol{\theta})$, and we can compute the point estimate of the parameters as $\boldsymbol{\theta}_k^* = \arg\min_{\boldsymbol{\theta}} \bar{\ell}_k(\boldsymbol{\theta})$. However, in our Bayesian FL setting, we will compute the (approximate) posterior distribution for each client using variational inference, which amounts to solving the following optimization problem

$$q_k^*(\boldsymbol{\theta}) = \arg\min_{q_k(\boldsymbol{\theta})} \mathcal{L}_k(q). \text{ where } \mathcal{L}_k(q) = \mathbb{E}_{q_k(\boldsymbol{\theta})}[\bar{\ell}_k(\boldsymbol{\theta})] + \mathbb{D}_{KL}(q_k(\boldsymbol{\theta}) \| p_k(\boldsymbol{\theta})). \tag{2}$$

where $p_k(\boldsymbol{\theta})$ is the prior and $\mathbb{D}_{KL}$ is the Kullback-Leibler divergence. If we use the Gaussian variational family for $q_k(\boldsymbol{\theta})$ with diagonal covariance then $q_k(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}_k, \text{diag}(\boldsymbol{\sigma}_k^2))$, where $\boldsymbol{m}_k$ and $\boldsymbol{\sigma}_k^2$ denote the variational parameters that are to be optimized for. Optimizing the objective in Equation 2 w.r.t these variational parameters requires making the following updates

$$\boldsymbol{m}_k^{t+1} = \boldsymbol{m}_k^t - \alpha \hat{\nabla}_{\boldsymbol{m}_k} \mathcal{L}_k(q);\ \boldsymbol{\sigma}_k^{t+1} = \boldsymbol{\sigma}_k^t - \alpha \hat{\nabla}_{\boldsymbol{\sigma}_k} \mathcal{L}_k(q) \tag{3}$$

where $\alpha > 0$ is the learning rate.

Computing exact gradients in the above update equations is difficult due to the expectation term in $\mathcal{L}_k(q)$. A naïve way to optimize is to use stochastic gradient estimators. However, these approaches are not very scalable due to the high variance in the gradient estimates. Shen et al. [20] improved these update equations and provided much more efficient update equations similar to Adam optimizer, which is essentially the improved variational online Newton (IVON) algorithm [20], with almost exact computational cost as Adam, and their key differences are summarized below

- Unlike Adam which solves for $\boldsymbol{\theta}$, IVON solves for both the mean vector $\boldsymbol{m}$ and the variances $\boldsymbol{\sigma}^2$ which provides us an estimate of the Gaussian variational approximation at each client. Note that the mean $\boldsymbol{m}$ plays the role of $\boldsymbol{\theta}$ in Adam. In addition, the variances naturally provide the uncertainty estimates for $\boldsymbol{\theta}$, essential for Bayesian FL (both in estimating the client models' uncertainties as well as during the aggregation of client models at the server).
- Unlike Adam which uses squared minibatch gradients to adjust the learning rates in different dimensions, IVON uses a reparametrization defined as gradient element-wise multiplied by $(\boldsymbol{\theta} - \mathbf{m})/\boldsymbol{\sigma}^2$ to get an unbiased estimate of the (diagonal) Hessian. Using this, IVON is able to get a cheap estimate of the Hessian, which makes it a second-order method unlike Adam.

2.2. Posterior aggregation at server

At the server, we can aggregate the client posteriors to compute the global posterior [9]. IVON approximates clients' posteriors as Gaussians and product of Gaussian distributions is still a Gaussian distribution up to a multiplicative constant. Thus we approximate the global distribution as a Gaussian whose optimal mean and covariance matrix expressions are given below. Moreover, since each client's variational approximation is a Gaussian with diagonal covariance matrix, it makes the aggregation operations efficient. Let's assume $q(\boldsymbol{\theta} \mid \mathcal{D}_k) = \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1})$ where $\boldsymbol{\mu}_k = \mathbf{m}_k$ and $\boldsymbol{\Lambda}_k = \text{diag}(\boldsymbol{\sigma}_k^2)$. Using results of the product of Gaussians based aggregation [9, 16], we have

$$\begin{aligned}
\log q(\boldsymbol{\theta} \mid \mathcal{D}) &\approx \sum_{k=1}^{K} w_k \log q(\boldsymbol{\theta} \mid \mathcal{D}_k) && \text{where } w_k = \frac{N_k}{\sum_{k=1}^{K} N_k} \\
q(\boldsymbol{\theta} \mid \mathcal{D}) &\approx \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) \\
\boldsymbol{\Lambda} &= \sum_{k=1}^{K} w_k \boldsymbol{\Lambda}_k && \boldsymbol{\mu} = \boldsymbol{\Lambda}^{-1} \sum_{k=1}^{K} w_k \boldsymbol{\Lambda}_k \boldsymbol{\mu}_k
\end{aligned} \tag{4}$$

Other aggregation strategies are also possible [9] and we leave this for future work. Note that our aggregation strategy can also be seen as Fisher-weighted model merging [8] where each client model is represented as the mean weights $\boldsymbol{m}_k$ and a Fisher matrix which depends on local posterior's

variances $\boldsymbol{\sigma}_k^2$ (although model merging only computes the mean, not the covariance, and thus does not yield a global posterior distribution at the server).

The appendix provides further details of IVON and its integration in our Bayesian FL setup, along with the pseudo-code for the overall algorithm, which we refer to as FedIvon.

Notably, FedIvon is appealing from two perspectives: It can be viewed an an efficient Bayesian FL algorithm offering the various benefits of the Bayesian approach, as well as a federated learning algorithm that easily incorporates second-order information during the training of the client models, while not incurring the usual overheads of second-order methods used by some FL algorithms [5].

3. Experiments

We experiment on three publicly available datasets: EMNIST [7], SVHN [19] and CIFAR-10 [14] (details in the appendix). We evaluate FedIvon in a challenging and realistic scenario involving heterogeneous data distribution among a large number of clients with each client having very few training examples. For each experiment, we consider a total of 200 clients with each client having a small private training set of less than 100 examples. To simulate non-iid data distribution, we randomly sample inputs from the training split, partition the sampled inputs into shards, and distribute shards among clients to create class-imbalanced training data similar to [6]. For a fair comparison, we use the same non-iid data split across clients for all the baseline methods and FedIvon. We follow the experimental setup of [4] and train customized CNN models on EMNIST, SVHN, and CIFAR-10 datasets. We compare our proposed method FedIvon with FedAvg [18] (simple aggregation of client models at server) and FedLaplace [16] (using the Laplace's approximation to fit a Gaussian distribution to each client's local model followed by aggregation at the server). FedAvg serves as a baseline to emphasize the importance of uncertainty quantification without compromising on the performance while FedLaplace serves as a competitive baseline to evaluate FedIvon's predictive uncertainty measures. For all the baselines and FedIvon, we run the federated algorithm for 2000 communication rounds, selecting a randomly sampled 5% i.e., 10 clients per round. We train each client's model locally for 2 epochs using a batch size of 32. We provide further details on hyperparameters, model architectures, and split in the appendix.

3.1. Classification Task

We train a classification model in FL setting using all the methods and report the results in Table 1. We evaluate all trained models' performance (accuracy and negative log-likelihood) on the test split and use metrics such as Expected Calibration Error (ECE) and Brier score to quantify predictive uncertainty. In our results, FedIvon@mean denotes point estimate based predictions evaluated at the mean of IVON posterior and FedIvon corresponds to Monte Carlo averaging with 500 samples.

As shown in Table 1, FedIvon outperforms all the baselines and yields the best test performance and calibration scores. FedIvon leverages the improved variational online Newton method to approximate the Hessian by continuous updates throughout the training. We also show the convergence of all the methods on all the datasets in Figure 1. As observed, FedIvon exhibits slightly slower improvements in the early training phase as compared to other baselines but soon outperforms them owing to its improved Hessian approximation as training progresses. Moreover, unlike FedLaplace which fits Gaussian distribution to the client's model using Laplace approximation evaluated at MAP estimate, FedIvon approximates the Hessian over the entire course of its training, resulting in much better predictive uncertainty estimates. This establishes efficacy of FedIvon, a

Models	EMNIST				CIFAR-10				SVHN			
	ACC(↑)	ECE(↓)	NLL(↓)	BS(↓)	ACC(↑)	ECE(↓)	NLL(↓)	BS(↓)	ACC(↑)	ECE(↓)	NLL(↓)	BS(↓)
FedAvg	91.66	0.0405	0.3355	0.1303	62.25	0.0981	1.199	0.5191	82.14	0.0311	0.6857	0.2640
FedLaplace	91.33	0.0381	0.3255	0.1314	61.80	0.1072	1.233	0.5284	81.99	0.0211	0.6423	0.2627
FedIvon@mean	**93.14**	0.0349	0.2821	0.1075	**62.92**	0.0983	1.1500	0.5114	84.54	0.0241	0.5624	0.2256
FedIvon	93.09	**0.0188**	**0.2341**	**0.1019**	62.54	**0.0312**	**1.0790**	**0.5021**	**84.76**	**0.0148**	**0.5303**	**0.2210**

Table 1: Test accuracy(ACC), Expected Calibration Error (ECE), Negative Log Likelihood (NLL), and Brier Score (BS)

second-order method, in efficient modeling of model and predictive uncertainty but with the computational cost that is similar to that of any first-order optimization method.

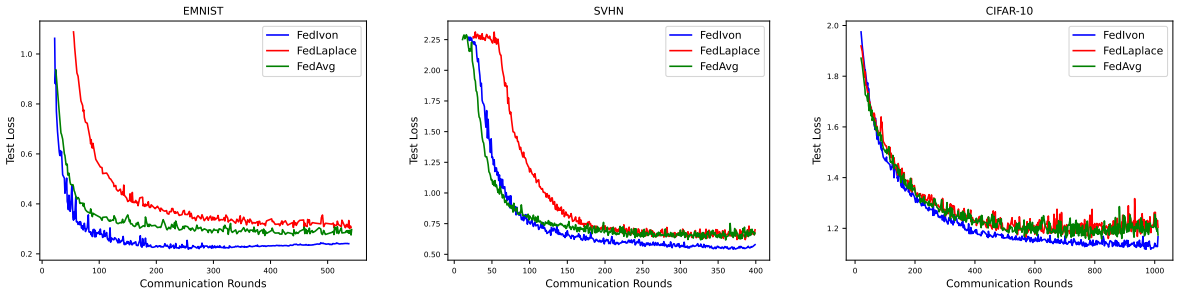

Figure 1: Convergence of various methods (left: EMNIST, center: SVHN, right: CIFAR-10).

3.2. Out-of-Distribution Detection Task

Predictive uncertainty of the model plays a crucial role in uncertainty-driven tasks such as OOD detection and active learning. We evaluate FedIvon and the baselines for distinguishing OOD inputs from in-distribution inputs using their predictive uncertainty. Given any input $\mathbf{x}$, the predictive uncertainty of the model's output is given by its Shannon entropy and is used to filter OOD inputs. We simulate this task by randomly sampling 5000 images from the OOD dataset and mixing it with an equal number of randomly sampled inputs from the test split of the training dataset.

Specifically, we use EMNIST, CIFAR-10, and SVHN as the OOD dataset for the models trained on EMNIST, SVHN, and CIFAR-10 respectively. We report the AUROC (area under the ROC curve) metric for all the methods on all the datasets in Table 2 which shows that FedIvon achieves better or competitive AUROC scores as compared to the other baselines.

Models	EMNIST	CIFAR-10	SVHM
FedAvg	0.8910	**0.7896**	0.7975
FedLaplace	0.8297	0.7513	0.8222
FedIvon	**0.9032**	0.7662	**0.8233**

Table 2: AUROC (↑) score for OOD/in-domain data detection

4. Conclusion

We presented an efficient Bayesian FL method which has a low training cost at each client (almost the same as running an Adam optimizer on each client) as well as low client-server communication cost. Furture work will investigate the effect of various aggregations strategies at the server, as well

as extending the method to personalized federated learning [1, 17, 21] where the data distribution exhibits significant heterogeneity across the different clients, further necessitating Bayesian FL.

References

[1] Idan Achituve, Aviv Shamsian, Aviv Navon, Gal Chechik, and Ethan Fetaya. Personalized federated learning with gaussian processes. *Advances in Neural Information Processing Systems*, 34:8392–8406, 2021.

[2] Maruan Al-Shedivat, Jennifer Gillenwater, Eric Xing, and Afshin Rostamizadeh. Federated learning via posterior averaging: A new perspective and practical algorithms. In *International Conference on Learning Representations*, 2020.

[3] Elaine Angelino, Matthew James Johnson, Ryan P Adams, et al. Patterns of scalable bayesian inference. *Foundations and Trends® in Machine Learning*, 9(2-3):119–247, 2016.

[4] Shrey Bhatt, Aishwarya Gupta, and Piyush Rai. Federated learning with uncertainty via distilled predictive distributions, 2023. URL https://arxiv.org/abs/2206.07562.

[5] Sebastian Bischoff, Stephan Günnemann, Martin Jaggi, and Sebastian U Stich. On second-order optimization methods for federated learning. *arXiv preprint arXiv:2109.02388*, 2021.

[6] Hong-You Chen and Wei-Lun Chao. Feddistill: Making bayesian model ensemble applicable to federated learning. *CoRR*, abs/2009.01974, 2020. URL https://arxiv.org/abs/2009.01974.

[7] Gregory Cohen, Saeed Afshar, Jonathan C. Tapson, and André van Schaik. Emnist: an extension of mnist to handwritten letters. *ArXiv*, abs/1702.05373, 2017. URL https://api.semanticscholar.org/CorpusID:12507257.

[8] Nico Daheim, Thomas Möllenhoff, Edoardo Maria Ponti, Iryna Gurevych, and Mohammad Emtiyaz Khan. Model merging by uncertainty-based gradient matching. *arXiv preprint arXiv:2310.12808*, 2023.

[9] John Fischer, Marko Orescanin, Justin Loomis, and Patrick McClure. Federated bayesian deep learning: The application of statistical aggregation methods to bayesian models. *arXiv preprint arXiv:2403.15263*, 2024.

[10] Han Guo, Philip Greengard, Hongyi Wang, Andrew Gelman, Yoon Kim, and Eric Xing. Federated learning as variational inference: A scalable expectation propagation approach. In *The Eleventh International Conference on Learning Representations*, 2023.

[11] Rahif Kassab and Osvaldo Simeone. Federated generalized bayesian learning via distributed stein variational gradient descent. *IEEE Transactions on Signal Processing*, 70:2180–2192, 2022.

[12] Mohammad Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and scalable Bayesian deep learning by weight-perturbation in Adam. In Jennifer Dy and

Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2611–2620. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/khan18a.html.

[13] Mohammad Emtiyaz Khan and Wu Lin. Conjugate-computation variational inference : Converting variational inference in non-conjugate models to inferences in conjugate models, 2017. URL https://arxiv.org/abs/1703.04265.

[14] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. URL https://api.semanticscholar.org/CorpusID:18268744.

[15] Florian Linsner, Linara Adilova, Sina Däubener, Michael Kamp, and Asja Fischer. Approaches to uncertainty quantification in federated deep learning. In *ECML PKDD Workshop on Parallel, Distributed, and Federated Learning*, pages 128–145. Springer, 2021.

[16] Liangxi Liu, Xi Jiang, Feng Zheng, Hong Chen, Guo-Jun Qi, Heng Huang, and Ling Shao. A bayesian federated learning framework with online laplace approximation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(1):1–16, January 2024. ISSN 1939-3539. doi: 10.1109/tpami.2023.3322743. URL http://dx.doi.org/10.1109/TPAMI.2023.3322743.

[17] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

[18] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[19] Yuval Netzer, Tao Wang, Adam Coates, A. Bissacco, Bo Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. 2011. URL https://api.semanticscholar.org/CorpusID:16852518.

[20] Yuesong Shen, Nico Daheim, Bai Cong, Peter Nickl, Gian Maria Marconi, Clement Bazan, Rio Yokota, Iryna Gurevych, Daniel Cremers, Mohammad Emtiyaz Khan, and Thomas Möllenhoff. Variational learning is effective for large deep networks, 2024. URL https://arxiv.org/abs/2402.17641.

[21] Xu Zhang, Yinchuan Li, Wenpeng Li, Kaiyang Guo, and Yunfeng Shao. Personalized federated learning via variational bayesian inference. In *International Conference on Machine Learning*, pages 26293–26310. PMLR, 2022.

Appendix A. Datasets

EMNIST consists of 28x28 grayscale images of alphabets and digits (0-9) with a train and test split comprising 124800 and 20800 images respectively; however, in our experiments, we restrict to alphabets only. SVHN consists of 32x32 RGB images of house number plates categorized into 10 distinct classes, each corresponding to one of the ten digits. It has a train and test split of size 73252 and 26032 respectively. CIFAR-10 comprises 32x32 RGB images of objects classified into 10 classes with 50000 training images and 10000 test images.

Appendix B. Hyperparameters used in experiments

In our experiments, We use ADAM optimizer with `learning_rate=1e-3, weight_decay=2e-4` for FedAvg and FedLaplace method. IVON[20] optimizer is used for FedIvon with different hyperparameters given in Table 3. Linearly decaying learning rate is used in all the experiments.

params	SVHN	EMNIST	CIFAR-10
initial learning rate	0.1	0.1	0.1
final learning rate	0.01	0.01	0.01
weight decay	2e-4	2e-4	2e-4
batch size	32	32	32
ESS (λ)	5000	5000	5000
initial hessian (h_0)	2.0	5.0	1.0
MC sample while training	1	1	1

Table 3: Ivon Hyperparameters

Appendix C. Additional Results on Convergence of FedIvon

Figure 2 shows the test accuracies of various methods with increasing number of rounds.

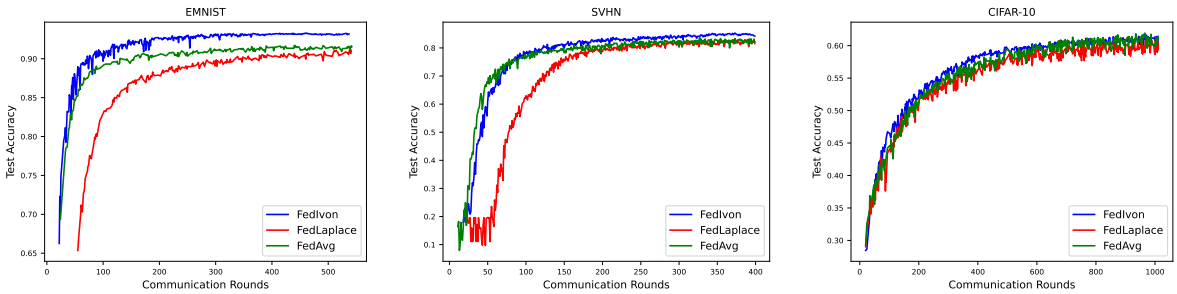

Figure 2: Test accuracy vs rounds (left: EMNIST, center: SVHN, right: CIFAR-10).

Algorithm 1 FedIvon Algorithm

Input: Total communication rounds R, total clients K, clients' private datasets $\{\mathcal{D}_k\}_{k=1}^{K}$, initial model weight $\boldsymbol{m_0}$, initial model Hessian $\boldsymbol{h}_0$

for $r = 1$ to R **do**
  Broadcast $\boldsymbol{m_r}, \boldsymbol{h_r}$ to all K clients
  Randomly sample k clients {*Update selected client models locally*}
  for $i = 1$ to k **do**
    $\boldsymbol{m_i}, \boldsymbol{h_i} =$ Client_Update$(D_i, \boldsymbol{m_r}, \boldsymbol{h_r})$
  end for
  Initialize $\boldsymbol{m_{r+1}} \leftarrow 0, \boldsymbol{h_{r+1}} \leftarrow 0$ {*Aggregation of client models at server*}
  for $i = 1$ to k **do**
    $\boldsymbol{h_{r+1}} \leftarrow \boldsymbol{h_{r+1}} + \boldsymbol{h}_i * w[i]$
    $\boldsymbol{m_{r+1}} \leftarrow \boldsymbol{m_{r+1}} + \boldsymbol{m}_i \odot \boldsymbol{h}_i * w[i]$
  end for
  $\boldsymbol{m_{r+1}} \leftarrow \frac{\boldsymbol{m_{r+1}}}{\boldsymbol{h_{r+1}}}$ (elementwise division) {*Global weight and Hessian*}
end for
Output: Global model weights and Hessian $(\boldsymbol{m_R}, \boldsymbol{h_R})$

Algorithm 2 Client_Update

Input: Local dataset D, model weights $\boldsymbol{m}$, Hessian($\boldsymbol{h}$), local_epochs(E), learning rates $\{\alpha_e\}$, weight decay δ, hyperparameters β_1, β_2, batch-size B
Output: Trained model weights $\boldsymbol{m}$, Hessian $\boldsymbol{\sigma}$

$\mathbf{g} \leftarrow 0, \quad \lambda \leftarrow |D|, n = E * |D|/B.$
$\boldsymbol{\sigma} \leftarrow 1/\sqrt{\lambda(\mathbf{h} + \delta)}.$
$\alpha_e \leftarrow (h + \delta)\, \alpha_e$ for all $e \in \{1, 2, \ldots, n\}$.
for $e = 1$ to E **do**
  Sample a batch of inputs of size B from D.
  $\widehat{\mathbf{g}} \leftarrow \widehat{\nabla}\bar{\ell}(\boldsymbol{\theta})$, where $\boldsymbol{\theta} \sim q$
  $\widehat{\mathbf{h}} \leftarrow \widehat{\mathbf{g}} \cdot (\boldsymbol{\theta} - \mathbf{m})/\boldsymbol{\sigma^2}$
  $\mathbf{g} \leftarrow \beta_1 \mathbf{g} + (1 - \beta_1)\, \widehat{\mathbf{g}}$
  $\mathbf{h} \leftarrow \beta_2 \mathbf{h} + (1 - \beta_2)\, \widehat{\mathbf{h}} + \frac{1}{2}\,(1 - \beta_2)^2\, (\mathbf{h} - \widehat{\mathbf{h}})^2/(\mathbf{h} + \delta)$
  $\overline{\mathbf{g}} \leftarrow \mathbf{g}/\,(1 - \beta_1^e)$
  $\mathbf{m} \leftarrow \mathbf{m} - \alpha_e(\overline{\mathbf{g}} + \delta \mathbf{m})/(\mathbf{h} + \delta)$
  $\boldsymbol{\sigma} \leftarrow 1/\sqrt{\lambda(\mathbf{h} + \delta)}$
end for

Appendix D. Details of IVON and Related Methods

Using natural gradients, Khan and Lin [13] gave improved gradient based update equations for the variational parameters and they call this approach Natural Gradient VI (NGVI). The major difference between NVGI and original update equations is that learning rate is now adapted by the variance $\boldsymbol{\sigma}_{\boldsymbol{k}}^{\boldsymbol{t+1}}$ which makes these updates similar to Adam.

$$\textbf{NVGI: } \boldsymbol{m}_k^{t+1} = \boldsymbol{m}_k^t + \beta^t \boldsymbol{\sigma}_k^{2^{t+1}} \odot [\hat{\nabla}_{\boldsymbol{m}_k} \mathcal{L}_k(q)];\ \boldsymbol{\sigma}_k^{-2^{t+1}} = \boldsymbol{\sigma}_k^{-2^t} - 2\beta^t [\hat{\nabla}_{\boldsymbol{\sigma}_k^2} \mathcal{L}_k(q)]$$

Further, Khan et al. [12] showed that the NVGI update equations can be written in terms of scholastic gradient and Hessian of $\boldsymbol{\theta}$, where $\boldsymbol{\sigma}_{\boldsymbol{k}}^{\boldsymbol{2}^t} = [N(\boldsymbol{h}_k^t + \lambda)]^{-1}$. The vector $\boldsymbol{h}_k^t$ contains an online estimate of diagonal Hessian. This approach called Variational Online Newton (VON) is similar to NGVI except that it does not require the gradients of the variational objective.

$$\textbf{VON: } \boldsymbol{m}_k^{t+1} = \boldsymbol{m}_k^t - \beta^t \frac{\hat{\boldsymbol{g}}(\boldsymbol{\theta}^t) + \lambda \boldsymbol{m}_k^t}{\boldsymbol{h}_k^{t+1} + \lambda};\ \boldsymbol{h}_k^{t+1} = (1-\beta^t)\boldsymbol{h}_k^t + \beta^t \text{diag}[\hat{\nabla}^2_{\boldsymbol{\theta}\boldsymbol{\theta}} \bar{\ell}_k(\boldsymbol{\theta}^t)]$$

In the update of VON for non-convex objective functions, the Hessian can be negative which might make $\boldsymbol{\sigma}_{\boldsymbol{k}}^{\boldsymbol{t}}$ negative, and break VON. To mitigate this issue Khan et al. [12] used a Generalized Gauss-Newton (GGN) approximation of Hessian which is always positive. This method is called VOGN.

$$\nabla^2_{\theta_j \theta_j} \bar{\ell}_k(\boldsymbol{\theta}^t) \approx \frac{1}{M} \sum_{i \in \mathcal{M}} \left[\nabla_{\theta_j} \ell_k^i(\boldsymbol{\theta}^t)\right]^2 := \hat{h}_j(\boldsymbol{\theta})$$

$$\textbf{VOGN: } \boldsymbol{m}_k^{t+1} = \boldsymbol{m}_k^t - \beta^t \frac{\hat{\boldsymbol{g}}(\boldsymbol{\theta}^t) + \lambda \boldsymbol{m}_k^t}{\boldsymbol{h}_k^{t+1} + \lambda};\ \boldsymbol{h}_k^{t+1} = (1-\beta^t)\boldsymbol{h}_k^t + \beta^t \hat{h}_j(\boldsymbol{\theta}^t)$$

VOGN [12] improves these equations where Gauss Newton estimation is used instead of Hessian which gives similar update equations as the Adam optimizer. However, it still uses per-sample squaring which is costly as compared to Adam. Further IVON [20] improved these update equations to make them as cost-effective as Adam. Using IVON's update equations, we can efficiently compute the client's local posterior $q_k^*(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta} | \boldsymbol{m}_k^*, \text{diag}(\boldsymbol{\sigma}_k^{*2}))$.

$$\textbf{IVON: } \widehat{\mathbf{h}}_k^t = \widehat{\nabla} \bar{\ell}_k(\boldsymbol{\theta}) \cdot \frac{\boldsymbol{\theta} - \mathbf{m}_k^t}{\boldsymbol{\sigma}_{\boldsymbol{k}}^{\boldsymbol{2}^t}};\ \mathbf{h}_k^{t+1} = (1-\rho)\mathbf{h}_k^t + \rho \widehat{\mathbf{h}}_k^t + \frac{1}{2}\rho^2 (\mathbf{h}_k^t - \widehat{\mathbf{h}}_k^t)^2 / \left(\mathbf{h}_k^t + s_0/\lambda\right)$$