# UNCERTAINTY QUANTIFICATION IN FEDERATED LEARNING

Deriving Fl-Swag as a one-shot method for the sake of calibration and privacy protection.

March 25, 2022

**Mehdi Makni**

**Advisors: Mérouane Debbah - Éric Moulines**

ÉCOLE POLYTECHNIQUE

IP PARIS

## Abstract

In this Bachelor Thesis report, we study one-shot methods whose objective is to obtain a well-calibrated model that results from a unique consensus step of the parameters of models trained in a federated fashion. To this end, we introduce uncertainty quantification and the calibration scores known in the field as well as the constraints of the Federated Learning setting. We motivate the problem of finding well-calibrated models that respect privacy issues encountered in Federated Learning and explore the performance of Swag, a rising star in uncertainty quantification, by comparison to the results of a ground-truth Hamiltonian Monte Carlo sampling method which is prohibitively expensive in the context of Deep Learning. We finally derive a highly-efficient, Swag-inspired, last-layer model that trains in a distributed way to allow clients to collaborate and solve a machine learning task without data sharing, while ensuring the calibration of their outputs. The results of the experiments are performed on MNIST, FashionMNIST and CIFAR 10 datasets and benchmarked against leading models in uncertainty quantification like Sgld and pSgld.

# Uncertainty Quantification in Federated Learning

## 1 Introduction

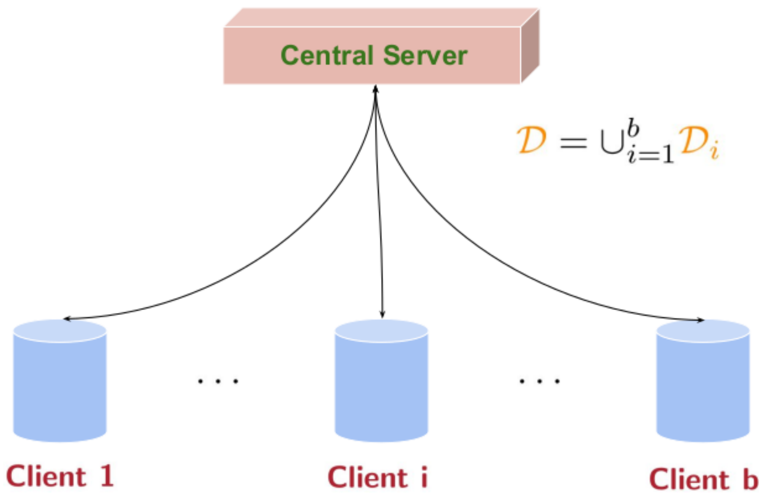
Uncertainty quantification is crucial in Deep Learning and we need calibrated models in order to make reliable decisions. It is known in the literature that Deep Learning methods tend to be overconfident and such predictions can be harmful or offensive (Amodei et al., 2016). Since uncertainty management is critical in many sectors, it is essential to have models that do well on known calibration scores to evaluate the decision process.
The quest of calibrated models is desirable because the outputs of machine learning methods can only be interpreted as probabilities if they are well-calibrated. Examples where estimating probabilities is critical includes sensitive domains where human life or large sums of money are at stake like healthcare, autonomous vehicles and financial markets.
These examples also share one more common feature in the fact that data could be sensitive and data sharing could be complicated especially due to government policies restrictions on data transfers (van der Burg et al., 2021). In particular, centralized storage, which is usually the case for machine learning applications, represents a data security problem since attackers only need to breach the central server to threaten the confidentiality of the whole database, making it a sensitive single point of failure (SPOF) system. There have been some work to overcome this issue by using cryptography-based technologies (Ji et al., 2014; Minelli, 2018). However, such approaches require massive communication costs between the clients and the server. Thus, there has been rising interest in Federated Learning, an approach where clients collaborate in solving a Machine Learning problem under the coordination of a central server or service provider and whose aim is to minimize the cost of the server-clients communication. One major concern in Federated Learning is privacy as we aim to run machine learning models on personal data without allowing for data exchange between devices nor with the server.
There are many application areas where Federated Learning as well as uncertainty representation are demanded. For this reason, we aim to tackle this issue and try to propose variants of leading algorithms in Bayesian Inference that are adapted to the setting of Federated Learning.
One natural candidate for such an algorithm would be the SWAG algorithm proposed by Maddox et al. (2019), which is a simple, scalable, and general purpose approach for uncertainty representation and calibration in deep learning. It has been shown that it performs well on a variety of tasks such as calibration, online learning and transfer learning.

**Paper Outline** The rest of the report is organized as follows: Section 2 goes into further details about Federated Learning, its constraints and the optimization paradigms one should aim for. Section 3 introduces uncertainty representation and MCMC algorithms, the issue of sampling from the true posterior distribution and why it is desirable for calibrated models. In Section 4, we motivate the SWAG algorithm as a starting method for both its practical and theoretical properties and we build on that to derive a one-shot method (Guha et al., 2019) that we denote by FL-SWAG in Section 5, an algorithm that fits a Gaussian on each client's local dataset and that aggregates the model weights obtained in the server. Section 6 is dedicated for a few experiments to better visualize sampling using SWAG against HMC on the Wisconsin Breast Cancer dataset. Throughout Section 7, we conduct multiple experiments and show that FL-SWAG performs very well when the problem is strictly convex, as in the case of Logistic Regression, but does not generalize well for Deep Learning tasks when the loss function's surface is not necessarily convex. To overcome this problem, we treat complex neural network architectures as a feature engineering step followed by a Classification via vanilla Logistic Regression step in Section 8. We allow the server to share the preprocessing step with the clients. Now the task for each client boils down to a Logistic Regression whose input is a highly-sophisticated preprocessed data. We finally benchmark the results obtained against Stochastic Gradient Langevin Dynamics SGLD (Welling and Teh, 2011) and a variant of the algorithm denoted by preconditioned Stochastic Gradient Langevin Dynamics pSGLD (Li et al., 2016).

## 2 Federated Learning

### 2.1 Introduction

Federated Learning is a distributed approach for collaborative machine learning from decentralized data (Wang et al., 2021). The clients collaborate in learning a global model thanks to communication rounds with the server, during which they do not share their local datasets, denoted $D_i$ for the $i^{th}$ client, but rather the gradient of their local loss function or similar data (Vono et al., 2021). A central motivation for Federated Learning is privacy protection. Indeed storing the data locally, for instance in mobile devices, compared to storing it in the cloud is a desirable approach because it reduces the attack surface of the system and the risk of breaching the whole database.



Federated Learning has seen increasing interest among researchers in academia as well

as in industry. As a matter of fact, many surveys on the subject of federated learning exist (Kairouz et al., 2019; Verbraeken et al., 2020; Wang et al., 2021). In practise, Federated Learning has been used in different contexts. For example Google makes use of Federated Learning in the eGboard mobile keyboard for application including next word prediction (Hard et al., 2019), emoji suggestion (Ramaswamy et al., 2019) and out-of-vocabulary word discovery. Similarly, Apple uses Federated Learning in iOS 13 for QuickType (Kairouz et al., 2021) or the vocal detection "Hey Siri".
We now try to introduce the core concepts in Federated Learning going through the constraints of this approach, the formalization of the optimization problems and the evaluation metrics.

## 2.2 Constraints

The approach of Federated Learning as initially proposed by McMahan et al. (2017) carries multiple constraints for it tries to deal with the problem of mobile devices that have a wealth of data suitable for learning models which can improve the user experience on the device but which tend to be privacy sensitive. This kind of data could include personal images, voice clips and text messages. Besides, the devices' content tend to differ greatly from one user to another depending on the user's preferences, the device's hardware and the time spent using the device. The setting of Federated Learning has the following characteristics:

**Non-IID** The training data for each client highly depends on the person using the device. Thus any particular user's local dataset is not representative for the entire population. In the problem formulation, we sometimes refer to this as statistically heterogeneous as training data may come from different distributions depending on the client.

**Imbalanced** Similarly, the size of the training data varies across users depending on how heavy they are going to use the device compared to the other users.

**Limited Availability** The mobile devices are frequently offline and it is unlikely they would be available to participate in each optimization round. This implies that in our optimization algorithms, the server can only access a small portion of the clients via some given sampling process without a guarantee that it will be able to communicate with them.

**Computational Constraints** The clients' devices usually have varying computing resources due to hardware differences. The differences can also range from computational speed to memory capacity. That's why we should take into account that the machine learning algorithms could be trained on usual mobile devices and should not be computationally heavy.

**Data Privacy constraints** For privacy reasons, the local datasets of each user cannot be shared with the server or communicated with other client devices.

## 2.3 Problem Formulation

### Federated Learning Cases

As has been introduced in detail in the work of Wang et al. (2021), the idea of Federated Learning usually takes place within the context of two different scenarios and this changes the way we design practical algorithms.

**Cross-device Federated Learning** This is the case where we deal with large populations of mobile devices. In this case, the number of devices could go beyond millions and the variability in terms of the above-mentioned constraints is higher across devices.

**Cross-silo Federated Learning** This is the case where we deal with as little as two and up to a few dozens of devices. For example, this could be the setting where different hospitals across the country collaborate on a machine learning task. The variability among devices is usually less significant than in the cross-device setting.

### Optimization Paradigms

The literature focuses on three main paradigms:

**Limiting the number of participating clients per round** In this paradigm, we try to allow only a limited small subset of the available clients to communicate with the server with the possibility to take into account a delay for non-participating workers which is motivated by the Limited Availability constraint of clients: (Langford et al., 2009; Agarwal and Duchi, 2011; Vono et al., 2021)

**Introducing Compression schemes during each Round** In this paradigm, we aim to compress the transmitted information (Agarwal et al., 2018) so as to lower the cost of communication between the clients and the server.

**Reducing the number of Communication Rounds** In this paradigm, we simply try to minimize the number of communication rounds between clients and the server for example by running as many local steps as possible before sharing model parameters with the server (Deng et al., 2021).

In this report we will mainly be working on the third paradigm and going to the extreme, that is we consider one-shot methods that communicate with the server only once and do a final aggregation step of models trained on the clients' private datasets.

## 2.4 Methods in the Literature and their Drawbacks

The literature of machine learning has seen some Bayesian models implemented in a distributed or federated approach to represent uncertainty. There has been many progress but most approaches (Liu and Simeone, 2021; Kassab and Simeone, 2021; Chen and Chao, 2021) tend not to take all the above mentioned constraints about Federated Learning into consideration or have no explicit theoretical bounds on the running time of such methods which puts us at risk of having a computationally expensive solutions.

## 3 Uncertainty Representation in Deep Learning

Bayesian methods present themselves as a suitable framework compared to good point estimate approches to Deep Learning as they lead to better accuracy and well-calibrated uncertainties. These methods allow to sample weights targeting a specific distribution but it is difficult to know which type of algorithm yields results closest to the true posterior distribution (Izmailov et al., 2021), since deep learning resides in a strongly non-convex problem, most theoretical safeguards are inapplicable. Indeed, the metrics classically used in optimization such as the accuracy or the loss evolution are not suitable to reflect how well an algorithm efficiently samples the posterior distribution. Wilson et al. (2021) are interested in determining the efficiency of classical methods for estimating the posterior distribution.
For a given data point $(x, y)$ where $x$ is the covariate and $y$ is the associated label, we aim to obtain the posterior predictive distribution:

$$p(y|x, \mathcal{D}) = \int p(y|x, \theta)p(\mathrm{d}\theta|\mathcal{D}).$$

The posterior predictive distribution computed by different methods is compared to the one obtained when the samples are provided by Hamiltonian Monte Carlo sampler which is known to be an asymptotically exact method (Betancourt, 2017), but tends to be excessively expensive in the context of deep learning and some recent experiments in other papers required hundreds of TPUs to run.

### 3.1 Calibration scores

To assess the performance of a neural network in terms of uncertainty quantification, we compute some calibration scores introduced in the paper of Guo et al. (2017). We now go through the definition and meaning of each score.

**Expected Calibration Error (ECE)** It measures, the difference between the accuracy and the confidence of the predictions model.

$$\mathrm{ECE} \approx \mathbb{E}_{(x,y)}\Big[\big|\mathbb{P}\left(y_{\mathrm{pred}}(x) = y \mid p(y_{\mathrm{pred}}(x)|x)\right) - p(y_{\mathrm{pred}}(x)|x)\big|\Big].$$

In practise, in order to estimate ECE we follow this procedure. We subdivide the interval $[0, 1]$ into $M$ sub-intervals. Then we define the bins $B_m = \{(x, y) \mid p(y_{\mathrm{pred}}(x)|x) \in [m - 1/M, m/M)\}$ for $m \in [M]$.

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{\mid B_m \mid}{\mid \mathcal{D}_{\mathrm{test}} \mid} \mid \mathrm{acc}(B_m) - \mathrm{conf}(B_m) \mid .$$

where $\mathrm{acc}(B_m)$ is the empirical accuracy and $\mathrm{conf}(B_m)$ is the average value of the confidence $p(y_{\mathrm{pred}}(x)|x)$ of the data points in the bin $B_m$.

**Brier Score** The Brier score can only be applied to random variables taking a finite number of values. BS computes the model's confidence in its predictions and is defined as follows:

$$\mathrm{BS} \approx \mathbb{E}_{(x,y)}\left[\sum_{c\in\mathcal{Y}}(p(y = c|x) - \mathbf{1}_{y=c})^2\right].$$

**Normalised negative log-likelihood (nNLL)** It helps measure the model ability to predict good labels with high probability.

$$\text{nNLL} \approx \mathbb{E}_{(x,y)}\left[ -\log p(y|x) \right].$$

**Next steps** Now that we have established the general setting of Federated Learning where we will be working and the metrics in the uncertainty representation part of the problem, we should now introduce the Swag algorithm and explore further how it will do against well-established models that account for uncertainty.

## 4 Original Swag Algorithm

### 4.1 Motivation

On March 31, 2021, there was a competition launched as a NeurIPS (Neural Information Processing Systems) event that aims to "Provide the best approximate inference for Bayesian Neural Networks according to a high-fidelity HMC reference". The details of the competition are explained in the paper of Wilson et al. (2021).
The ultimate goal from the competition is to promote research about methods that can deal with uncertainty representation which is crucial for the safe and reliable deplyoment of machine learning. The competition benchmarks the proposed algorithms against Hamiltonian Monte Carlo (HMC) (Brooks et al., 2011), a well-studied Markov Chain Mote Carlo (MCMC) method which is guaranteed to asymptotically produce samples from the true posterior but which tends to be prohibitively expensive in the context of deep learning.

Some variants of Swag were proposed for the competition and ended up in top positions. The algorithm is described by the authors Maddox et al. (2019) as simple, scalable, and general purpose approach for uncertainty representation and calibration in deep learning. It has been shown to perform well on multiple tasks including sample detection, calibration and transfer learning.

Another motivation to use Swag in the Federated Learning setting is that it fits a Gaussian to estimate the covariance matrix in the weight space for each client. Aggregating Gaussian distributions has desirable properties as will be discussed in Section 5 and present themselves as a natural approach for distributed machine learning algorithms.

### 4.2 Description

Maddox et al. (2019) described in the original paper of Swag that the algorithm starts with an initial weight $\theta_0$ from a pretrained solution. It then tries to explore the geometry of the weight space thanks to running Sgd iterations with a high learning rate schedule and tries to efficiently fit a covariance matrix near the averaged weight denoted $\theta_{\text{SWA}}$ which is derived from the trajectory of Sgd iterates.
Swag is inspired and builds on the method Swa which does not address the problem of uncertainty representation but rather aims to increase generalization performance.

In the paper of Izmailov et al. (2019), it has been argued that SGD iterations converge to the boundary of the set of high-performing solution whereas their method SWA which averages the values of weights of these iterations helps reduce generalization error. The weight found by this method is given by $\theta_{\text{SWA}} = \frac{1}{T}\sum_{i=1}^{T}\theta_i$
In order to run a sampling procedure within a Bayesian approach, the SWAG algorithm tries to derive from the SGD iterates, an approximate posterior distribution over the weights. The aim is to be able to sample from the Gaussian distribution to perform Bayesian model averaging. Some experiments demonstrate that SWAG approximates the shape of the true posterior well. In order to do that, SWAG builds on deriving information from the SGD trajectory to construct in a first phase a diagonal format of the covariance matrix by maintaining a moving average of the second uncentered moment of each weight. That is it obtains $\Sigma_{\text{diag}} = \text{diag}(\overline{\theta^2} - \theta^2_{\text{SWA}})$ where $\overline{\theta^2} = \frac{1}{T}\sum_{i=1}^{T}\theta_i^2$
SWAG then extend this to add a more flexible low-rank covariance matrix in a second phase $\Sigma \approx \frac{1}{T-1}\sum_{i=1}^{T}\left(\theta_i - \bar{\theta}_i\right)\left(\theta_i - \bar{\theta}_i\right)^{\top} = \frac{1}{T-1}DD^{\top}$, where $D$ is the deviation matrix with columns $D_i = \left(\theta_i - \bar{\theta}_i\right)$, and $\bar{\theta}_i$ is the running average of . To make sure the rank of the estimated covariance matrix is not too large which makes computations prohibitively expensive, we use the last $K$ of the $D_i$ vectors.
Training SWAG on a client therefore is explained in the following pseudocode (Maddox et al., 2019):

**Algorithm 1** Train-SWAG

**Inputs:** $\theta_0$: pretrained weights, $\eta$: learning rate, $T$: number of steps, $c$: moment update frequency, $K$: maximum number of columns in deviation matrix, $S$: number of samples in Bayesian model averaging.

```
$\overline{\theta} \leftarrow \theta_0, \overline{\theta^2} \leftarrow \theta_0^2$
for $i \leftarrow 1, 2, \dots, T$ do
    $\theta_i \leftarrow \theta_{i-1} - \eta\nabla_\theta\mathcal{L}(\theta_{i-1})$
    if MOD$(i, c) = 0$ then
        $n \leftarrow i/c$
        $\overline{\theta} \leftarrow \frac{n\overline{\theta}+\theta_i}{n+1}$, $\overline{\theta^2} \leftarrow \frac{n\overline{\theta^2}+\theta_i^2}{n+1}$
    if NUM_COLS$(\hat{D}) = K$ then
        REMOVE_COL$(\hat{D}[:, 1])$
    APPEND_COL$(\hat{D}, \theta_i - \overline{\theta})$
end for
return $\theta_{SWA} = \overline{\theta}, \Sigma_{diag} = \overline{\theta^2} - \overline{\theta}^2, \hat{D}$
```

Moreover, we can obtain the calibrated output using Bayesian Model Averaging for SWAG following this pseudocode (Maddox et al., 2019).

**Algorithm 2** Test: Bayesian Model Averaging

```
for $i \leftarrow 1, 2, \dots, S$ do
    Draw $\tilde{\theta}_i \sim \mathcal{N}\left(\theta_{SWA}, \frac{1}{2}\Sigma_{diag} + \frac{\hat{D}\hat{D}^T}{2(K-1)}\right)$
    Update batch norm statistics with new sample.
    $p(y^*|Data) \mathrel{+}= \frac{1}{S}p(y^*|\tilde{\theta}_i)$
end for
return $p(y^*|\text{Data})$
```

## 5 FL-Swag

### 5.1 Desirable Theoretical Results

We know so far that if we have $b$ clients, we could train SWAG on each of the clients individually. This of course happens locally on the devices without any data exchange which follows the constraints of Federated Learning.
We therefore obtain for $i \in [b], \widehat{\mu_i} = \theta_{\text{SWA}_i}, \widehat{\Sigma_i} = \frac{1}{2}\Sigma_{\text{diag }i} + \frac{\hat{D}_i \hat{D}_i^T}{2(K-1)}$ and hence a way to estimate each subposterior density with $\widehat{p}_i(\theta \mid \mathcal{D}_i) = \mathcal{N}_d\left(\theta \mid \widehat{\mu_i}, \widehat{\Sigma_i}\right)$. Note that thanks to a Monte Carlo sampling procedure, we can approximate:

$$p\left(y \mid x, \mathcal{D}\right) \approx \frac{1}{T}\sum_{t=1}^{T} p\left(y \mid x, \theta_t\right), \qquad \theta_t \sim p(\theta \mid \mathcal{D}).$$

We will assume that sharing such information with the server preserves the anonymity of the data stored locally. That is we assume that the topology of the loss function of the used model near the optimum does not hide information about the data used for training.
We now require the server to aggregate this data. Neiswanger et al. (2013) have shown that the calculations in this step turn out to be quite nice thanks to the Gaussian distributions properties. Indeed, The product of the $b$ subposterior densities will be proportional to a Gaussian pdf, and our estimate of the density product function $p_1 \cdots p_b(\theta) \propto p\left(\theta \mid x^N\right)$ is

$$\widehat{p_1 \cdots p_b}(\theta \mid \mathcal{D}) = \widehat{p}_1(\theta \mid \mathcal{D}_1) \cdots \widehat{p}_b(\theta \mid \mathcal{D}_b) \propto \mathcal{N}_d\left(\theta \mid \mu_S, \Sigma_S\right).$$

Note that we use the subscript $S$ to denote the Server's parameters.

$$\Sigma_S = \left(\sum_{m=1}^{b} \widehat{\Sigma}_i^{-1}\right)^{-1}.$$

$$\mu_S = \Sigma_S \left(\sum_{m=1}^{b} \widehat{\Sigma}_i^{-1}\widehat{\mu}_i\right).$$

### 5.2 FL-Swag Description

Thanks to the above mentioned desirable theoretical results of Gaussian distributions, we are in good shape to provide the pseudocode for FL-SWAG:

**Algorithm 3** FL-SWAG

```
for $i = 1 \to b$ do
    $\widehat{\mu}_i, \widehat{\Sigma_i} \leftarrow$ TrainSWAG(client$_i$)
end for
$\Sigma_S \leftarrow \left(\sum_{i=1}^{b} \widehat{\Sigma}_i^{-1}\right)^{-1}$
$\mu_S \leftarrow \Sigma_S \left(\sum_{i=1}^{b} \widehat{\Sigma}_i^{-1}\widehat{\mu}_i\right)$
return $\mu_S, \Sigma_S$
```

### 5.3 Online Learning with FL-Swag

Note that there is yet another nice property of working with Swag in a distributed fashion thanks to the properties of Gaussian distributions.
We have seen that the approach of training Swag on individual clients and then aggregating the results in the server is very desirable because there is only one communication round between clients and the server which reduces the threat surface of information leakage, a key property in our Federated Learning approach. The good news is that if one more client is yet to join the training group or even if a client wishes to update its distribution parameters, it is very easy and done with only one communication round with that client only and without any further interaction of previous clients.
Indeed assume, we have trained Swag on $b$ clients and the server now has parameters $\Sigma_S = (\sum_{i=1}^{b} \widehat{\Sigma}_i^{-1})^{-1}$ and $\mu_S = \Sigma_S(\sum_{i=1}^{b} \widehat{\Sigma}_i^{-1}\widehat{\mu}_i)$ and now we want to also include the parameters of one more client $b+1$ after the training is done whose parameters are given by: $\widehat{\mu}_{b+1}$ and $\widehat{\Sigma}_{b+1}$, the server simply updates his parameters as follows:

$$\Sigma_{S_{\text{new}}} \leftarrow \left(\Sigma_S^{-1} + \widehat{\Sigma}_{b+1}^{-1}\right)^{-1}.$$

$$\mu_{S_{\text{new}}} \leftarrow \Sigma_{S_{\text{new}}} \left(\Sigma_S^{-1}\mu_S + \widehat{\Sigma}_{b+1}^{-1}\widehat{\mu}_{b+1}\right).$$

Similarly, if a client $i \in [b]$ updates their parameters from $\widehat{\mu}_i$ to $\overline{\mu}_i$ and $\widehat{\Sigma}_i$ to $\overline{\Sigma}_i$, the server does the following:

$$\Sigma_{S_{\text{new}}} \leftarrow \left(\Sigma_S^{-1} - \widehat{\Sigma}_i^{-1} + \overline{\Sigma}_i^{-1}\right)^{-1}.$$

$$\mu_{S_{\text{new}}} \leftarrow \Sigma_{S_{\text{new}}} \left(\Sigma_S^{-1}\mu_S - \widehat{\Sigma}_i^{-1}\widehat{\mu}_i + \overline{\Sigma}_i^{-1}\overline{\mu}_i\right).$$

Note that this case could be very common in the setting of Federated Learning because a client could change his general use of the device. Examples could include a change in behavior, personality, preferences and financial situation.

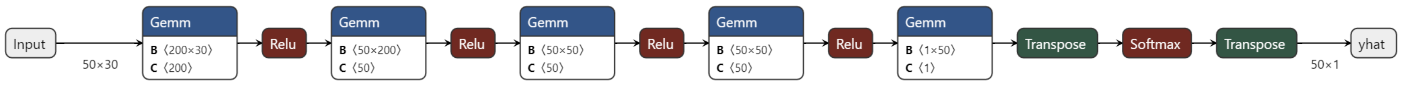
## 6 Toy Examples

### 6.1 Swag-HMC toyish Comparison

We start by building a toy example and try to visualize and quantify to what extent can the Swag algorithm be close to Hamiltonian Monte Carlo method which samples asymptotically from the true posterior distribution.

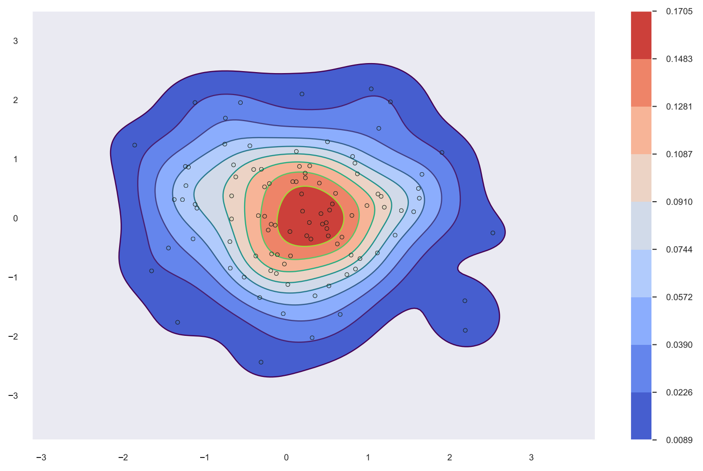
**Dataset** For a toy example, we use the Breast Cancer Wisconsin dataset which could be easily accessible from scikit-learn. This dataset is easy to work with because the input dimension is 30 and the output dimension is 1. Besides, healthcare data represents a very suitable field where we want to have calibrated models that respect the Federated Learning constraints. For example, we can imagine hospitals that have locally the database of their patients which should not be communicated with other hospitals or medical professionals but they do indeed want to have calibrated models whose outputs can be interpreted as probabilities.

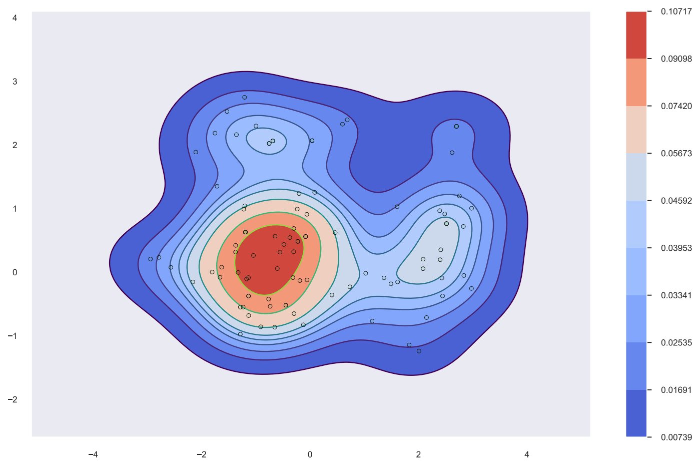
**Model** To deal with this dataset, we build a simple feed-forward-neural network 6.1 that has 3 hidden layers and the following architecture with Gemm meaning General Matrix Multiplication (Dense Layer).



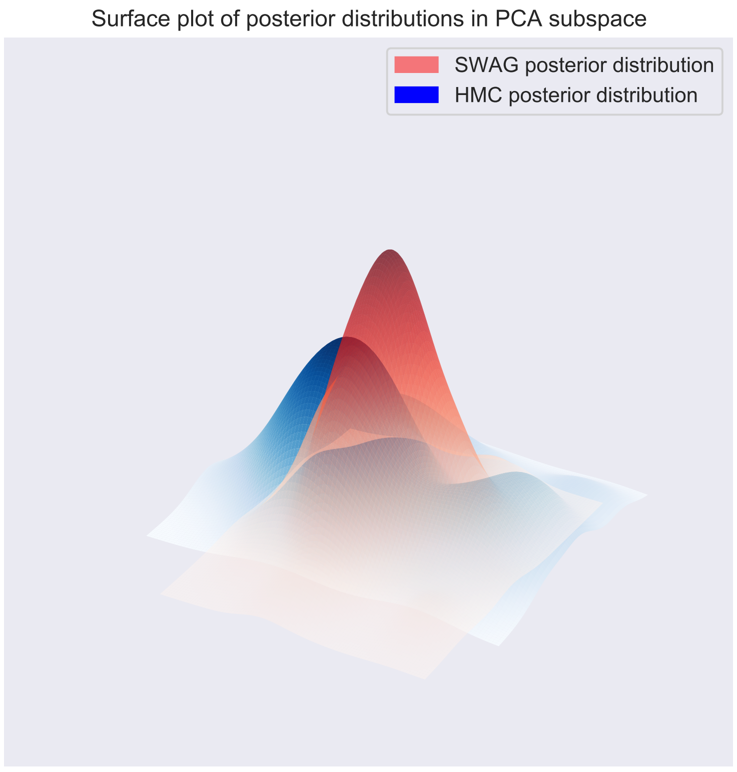
**Subspace Inference** We start by running SWAG algorithm on the provided dataset. We then try to benchmark the obtained samples of weights to the ground truth which would be the samples obtained from the Hamiltonian Monte Caro method (HMC). The samples obtained lie in a 21401-dimensional space. The average deviation on coordinates of the difference between the vectors obtained by SWAG and HMC in this high-dimensional space is of the order of $10^{-3}$. This indeed indicates that SWAG is doing a good job sampling from the true posterior distribution which is essential in uncertainty quantification.

**Visualization** In order to visualize how well SWAG is actually doing, we get inspired by the work of Izmailov et al. (2020); Garipov et al. (2018) and try to project the samples obtained from SWAG and HMC on a plane obtained thanks to PCA (Principal Component Analysis). The results obtained are reported in these figures. To the left, we have the SWAG samples projected in the 2D plane. To the right, we have the HMC samples projected in the sample 2D subspace.





As we can see, the projected distribution of SWAG is pretty close to that of HMC, and it has fitted a Gaussian that can partially explain the topology of HMC in this subspace. We now try to visualize the two plots in 3D for better visualization.



## 7 Experiments

### Datasets Used

In our experiments, we will use the standard datasets MNIST and FashionMNIST (Xiao et al., 2017) which represent a famous benchmark for machine learning models compromising 28 x 28 gray scale images with 60000 training images. The two datasets share the same image size, data format and the structure of training and testing splits. For a harder dataset, we take CIFAR 10 dataset (Krizhevsky, 2009), which consists of 60000 32x32 colour images (3 channels for RGB data) in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

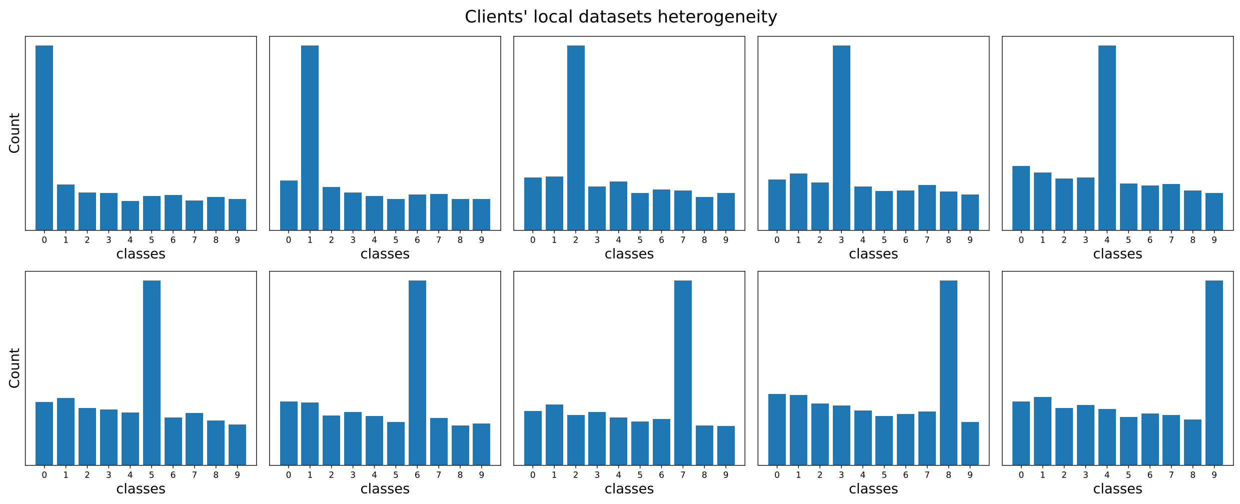
### Data Splitting

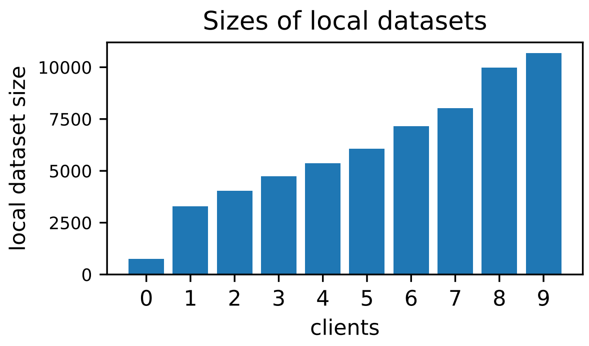
One key constraint in Federated Learning is that clients' datasets are non i.i.d. For this reason, we need to split the usual MNIST, FashionMNIST and later on preprocessed CIFAR 10 across clients in a way where we respect those constraints. To do this, we select a distribution $\mathcal{P}$, for example uniform in $[0, 1]$, we sample $n_{clients}$ weights and then we sort these samples in an increasing order. We then define a proportion value $p$ where client $c$ would be 'interested' in class $c \mod n_{classes}$, $1/p$ more than the rest of classes. We finally normalize the obtained matrix in $L1$. Note that by symmetry over clients as well as the classes justifies this approach. More concretely, we follow this method:

**Algorithm 4** noniid-datasets

```
  Input: $n_{clients}, \mathcal{P}, p$, train_loader
for $m = 1 \rightarrow n_{clients}$ do
    size$_m \sim \mathcal{P}$                                  ▷ Sample sizes from the distribution
end for
size.sort()                                             ▷ By symmetry, we can sort clients by size
for $m = 1 \rightarrow n_{clients}$ do
    for $c = 1 \rightarrow n_{classes}$ do
        if MOD$(c, m) = 0$ then                          ▷ Define heterogeneous interests for clients
            interest$_{m,c} \leftarrow 1$
        else
            interest$_{m,c} \leftarrow p$
        end if
    end for
end for
for $m = 1 \rightarrow n_{clients}$ do
    for $c = 1 \rightarrow n_{classes}$ do
        statistics$_{m,c} \leftarrow \frac{\text{size}_m \cdot \text{interest}_{m,c}}{\sum_{m',c} \text{size}_{m'} \cdot \text{interest}_{m',c}}$
    end for
end for
for $c = 1 \rightarrow n_{classes}$ do
    Split(train_loader, $c$, weights = statistics$_c$)            ▷ Split into client_loaders
end for
```

**Visualization** After running this splitting algorithm, for $b = 10$ as in all our experiments, we obtain, in figure one, the statistical distributions of the clients which confirms their heterogeneity and, in figure two, the size or number of images in the local datasets of the clients which confirms that the datasets are imbalanced.





## 7.1 FL-Swag with vanilla Logistic Regression

We first try to assess the performance of SWAG when the model has the property that the loss surface is strictly convex as in the case of Logistic Regression. For such an easy model, we will use the datasets MNIST and FashionMNIST.
FL-SWAG will be compared against multiple methods:

- FEDAVG (McMahan et al., 2017) which is a simple average of the weights of the model, in this case Logistic Regression.
- WFEDAVG which carries the same idea as the previous method but takes also into account the size of the dataset (gives more priority to clients' whose dataset is larger).
- BAYAVG This method asks every client to communicate their prediction and takes the mean of the aggregation.
- WBAYAVG Similar to the previous method but account for the size of each client's local dataset.

Note that these methods do not necessarily respect the Federated Learning constraints. For example, we do not wish to ask every client to share the probability of a new data point and it might also be a privacy issue to share the size of the dataset with the server.

**Training Swag** We train the SWAG algorithm on each client's local dataset. This allows us to obtain $\widehat{\mu_i}$ as well as $\widehat{\Sigma_i}$ for each client $i \in [b]$.
We now report the accuracy of each model on the testing dataset as well as the accuracy of the model obtained by FL-SWAG, FEDAVG, WFEDAVG, BAYAVG, WBAYAVG.

| Weights | Accuracy (MNIST) | Accuracy (FashionMNIST) |
|---|---|---|
| client 1 | 84.07% | 75.72% |
| client 2 | 87.41% | 80.15% |
| client 3 | 87.75% | 78.63% |
| client 4 | 89.57% | 80.81% |
| client 5 | 90.21% | 81.11% |
| client 6 | 90.09% | 81.75% |
| client 7 | 90.77% | 80.82% |
| client 8 | 90.29% | 82.51% |
| client 9 | 90.69% | 83.1% |
| client 10 | 90.8% | 82.8% |
| FL-Swag | 90.96% | 81.32% |
| FedAvg | 91.58% | 83.0% |
| wFedAvg | 91.93% | 83.63% |
| BayAvg | 91.65% | 83.05% |
| wBayAvg | 91.98% | 83.61% |
| CommonRes | 92% | 84% |

**Interpretations** In this Logistic Regression setting, we can see the FL-Swag is doing pretty well by outperforming most previous clients. Bayesian Model Averaging which consists of predicting the training data with the weights of each single client and taking the mean of the output is computationally expensive and in a federated learning approach, we do not wish to communicate with every client just to make a single prediction.

## 8 FL-Swag with Deep Neural Networks

Now that we have made sure FL-Swag has many desirable properties and respects well privacy constraints, we want to see how well the model obtained by the server in FL-Swag performs in the context of Deep Learning. We will compare it to other methods in Bayesian Inference that have been shown to perform well in uncertainty quantification and calibration. In the experiments, these methods will not be subject to the Federated Learning constraints but rather help to serve as a baseline of comparison on the metrics introduced by Guo et al. (2017).

### 8.1 Algorithms for Comparison

For the training of FL-Swag, we will use all the datasets in our experiments: "MNIST", "FashionMNIST" as well as "CIFAR 10". We will split a given dataset on $b = 10$ clients in such a way that it is imbalanced and statistically heterogeneous (as has been described in the Data Splitting paragraph in Section 7).
That way, the number of images, as well as the distribution of each label (there are 10 different labels in both considered datasets) can vary significantly between each client's private dataset. After that, FL-Swag is run as explained: firstly by training Swag on each client's local dataset and then by aggregating these results on the server.

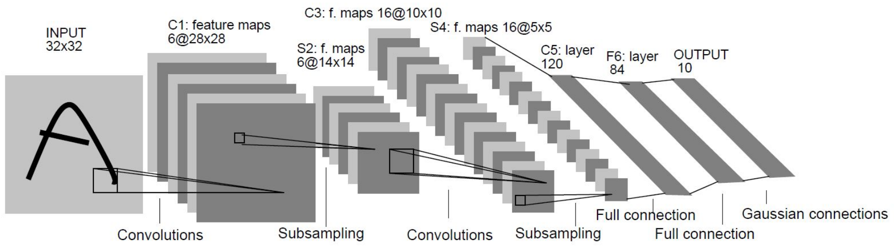
#### 8.1.1 Sgld and pSgld

Sgld or Stochastic Gradient Langevin Dynamics is a bayesian learning framework proposed by Welling and Teh (2011) that aims to add the right amount of noise to a standard stochastic gradient optimization algorithm usually Sgd with theoretical guarantees about the convergence of the iterates to sample from the true posterior

distribution.
Similarly, pSGLD or Preconditioned Stochastic Gradient Langevin Dynamics for Deep Neural Networks (Li et al., 2016) builds on SGLD to add adaptive preconditioners so that dealing with changing curvature is more efficient.
These two bayesian learning methods are well-known in the literature and have theoretical guarantees and practical successes as well. That is why we pick them as baseline methods.

## 8.2 FL-Swag and consensus step federated learning in Deep Learning

Since we have seen that FL-SWAG achieves what is expected from it in a strictly convex Logistic Regression setting, our aim is to take this idea one step further and try to train a deep neural network model, in our case LeNet5 8.2 (Lecun et al., 1998), on different client's heterogeneous and imbalanced local datasets as usual and try to aggregate the weights obtained in a final consensus step in the context of Deep Learning.



However, our experiments have shown that the results obtained are very poor and any kind of model Bayesian averaging seem not to work (Including FEDAVG and wFEDAVG). Further, they have demonstrated that in the setting of Deep Learning, models are more sensitive to change of weights compared to Logistic Regression (surface of loss function is strictly convex) and combining models does not seem to give good results. One idea that explains this in Deep architectures is the fact that permuting the neurons in a succession of dense hidden layers should theoretically give the same results after training. Indeed, Wang et al. (2020) have tried to overcome this issue within the context of Federated Learning by matching and averaging hidden neurons in the Deep Learning architectures and created the algorithm Federated Matched Averaging FEDMA.
These observations of Wang et al. (2020) confirm that averaging weights blindly in Deep Learning models does not make much sense, if not taken care of by some idea such as this best-matching scheme.

## 8.3 Introducing server preprocessing step

It has been shown in the work of Brosse et al. (2020) that there is limited value in adding multiple uncertainty layers to deep classifiers and that families of algorithms that split the task of classification into representation learning (or feature extraction) and uncertainty estimation strongly outperform vanilla point-estimate SGD and obtain good results on calibration scores.
Our next step is inspired by such approach and for this reason, we pretrain a model in the server and allow the latter to communicate initial layer weights, which represents

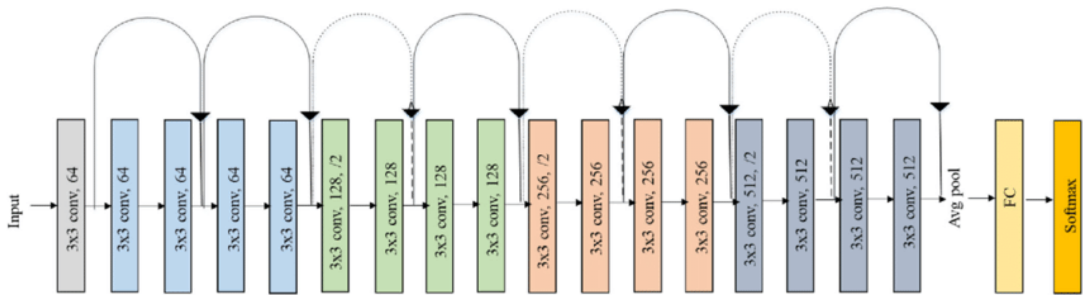
the feature extraction part, to the clients but not those of the last layer, which represents the uncertainty quantification step, and then let the clients retrain the model on their private datasets and finally share the resulting SWAG parameters with the server as in the usual approach.
In this matter, we are considering the following simpler way to compute the predictive posterior distribution and we think it should be enough to account for uncertainty based on the previous idea:

$$p(y \mid x, \mathcal{D}) = \int p(y|x, \theta_{\text{last\_layer}}) p(\mathrm{d}\theta_{\text{last\_layer}}|\mathcal{D}).$$

**Why it might work?** A Deep Neural Network architecture whose aim is classification commonly has for an output layer a Dense layer with shape compromising the size of the last hidden layer multiplied by the size of the output (Example LeNet5 8.2). This last output layer could be looked at as a Logistic Regression model that takes as input a highly sophisticated preprocessed data and thus the initial layers of the neural network represent a way to conduct the representation learning step.
The idea that we could build on from this observation is that a server can communicate to the clients at an initial step a black box method to preprocess the data they own. The clients would apply this feature extraction function shared by the server to their local datasets, which are still not shared with outer sources and thus respecting the Federated Learning approach.

**Freezing the initial layers** In order to see the effect of this idea, we conduct two separate experiments.
In the first one, conducted on both MNIST and FashionMNIST, clients start with a model whose initial layers are initialized as in the general LeNet5 8.2 pretrained model. We see through many experiments that the stability of the consensus step highly depends on the number of local steps performed on each client. The higher the number of local iterations, the more we observe the *client-drift* phenomena, where clients tend to overfit for their local datasets and deviate from the initial model. Therefore, the consensus step becomes less stable and we could get bad results if the number of local steps is big. The results are shown when clients train for 20 epochs 8.3.

In the second one, conducted on CIFAR 10, we create a preprocessing function that takes the weights of the ResNet18 model 8.3 (He et al., 2016) but not those of the last layer, and then we pass this function to each client.



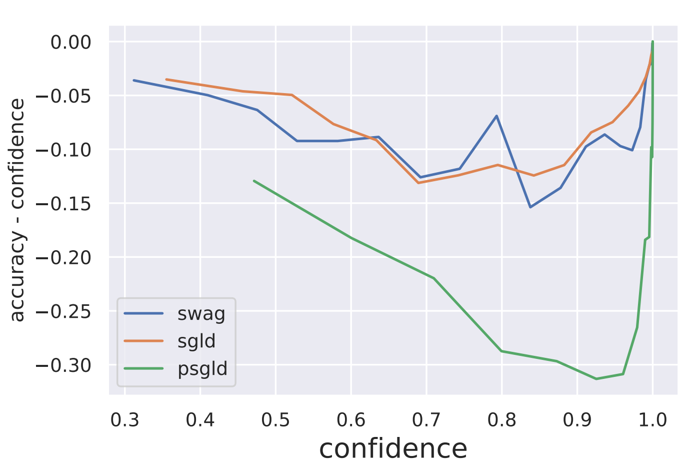
Every client then preprocesses the input data to convert each image from a 3x32x32 image (dimensions of a CIFAR 10 image) to a 512 dimensional vector (Size of the output of the last hidden layer in ResNet18 8.3) using this function. Later, they simply train Logistic Regression as in the initial experiments of this Section. We notice that this allows the consensus step to be very stable and the number of local iterations does not influence it as much as in the first experiment. The results are shown when clients train for 30 epochs. The accuracies obtained are reported in the table 8.3.

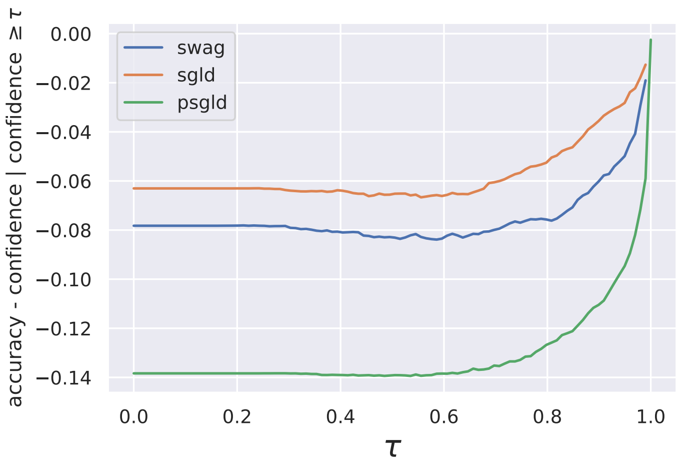
| Weights<br>Dataset<br>Model | Accuracy<br>MNIST<br>LeNet5 | Accuracy<br>FashionMNIST<br>LeNet5 | Accuracy<br>CIFAR 10<br>ResNet18 |
|---|---|---|---|
| client 1 | 97.74% | 86.47% | 69.85% |
| client 2 | 97.99% | 87.36% | 69.69% |
| client 3 | 98.03% | 85.53% | 69.73% |
| client 4 | 97.44% | 86.71% | 69.6% |
| client 5 | 98.02% | 85.66% | 70.29% |
| client 6 | 97.98% | 86.92% | 70.13% |
| client 7 | 98.18% | 85.85% | 70.18% |
| client 8 | 97.9% | 87.03% | 70.28% |
| client 9 | 97.85% | 87.26% | 70.46% |
| client 10 | 98.15% | 86.98% | 70.35% |
| FL-SWAG | 97.41% | 87.57% | 70.35% |
| FedAvg | 97.64% | 87.6% | 70.26% |
| wFedAvg | 97.44% | 87.48% | 70.44% |
| BayAvg | 98.57% | 87.58% | 70.32% |
| wBayAvg | 98.55% | 87.49% | 70.45% |

As we can see, this time FL-SWAG actually performed well on classification and the last consensus step makes sense. At this point, we have already obtained a model that respects Federated Learning constraints and that achieves state-of-the-art accuracy even though it aggregates weights from models whose datasets are statistically heterogeneous and imbalanced. Now, we want to validate their uncertainty quantification performance. For this reason, we compute their calibration scores and benchmark them against well-known models in the field.

| Dataset | Model | Method | Final accuracy | ECE x $10^2$ | BS x 10 | NLL x 10 |
|---|---|---|---|---|---|---|
| MNIST | LeNet5 | FL-SWAG | 97.41% | 0.77 | 0.41 | 0.86 |
| | | SGLD | 97.61% | 1.22 | 0.38 | 0.89 |
| | | pSGLD | 97.42% | 1.97 | 0.45 | 1.58 |
| FashionMNIST | LeNet5 | FL-SWAG | 87.57% | 6.92 | 3.18 | 4.69 |
| | | SGLD | 87.15% | 4.16 | 1.90 | 3.91 |
| | | pSGLD | 86.67% | 2.73 | 1.95 | 3.93 |
| CIFAR 10 | ResNet18 | FL-SWAG | 70.35% | 7.82 | 4.18 | 9.18 |
| | | SGLD | 75.16% | 6.38 | 3.51 | 7.39 |
| | | pSGLD | 77.67% | 13.84 | 3.55 | 9.77 |

The accuracy - confidence graph is shown for the CIFAR 10 experiment here 8.3:





The two plots represent calibration plots to visualize better the performance of these methods that account for uncertainty. The first one is the plot of the difference between accuracy and confidence versus the confidence value. The second one is the same difference when the confidence in predictions is greater than a threshold value $\tau$ versus this value $\tau$ that naturally ranges in the interval $[0, 1]$. More concretely, as in the work of Vono et al. (2021), we consider for $\tau \in [0, 1]$, the set $D^{(\tau)}_{\text{pred}} = \{x \in D_{\text{test}} : p(y|x) \geq \tau\}$ of classified data with credibility greater than $\tau$. Then we compute the accuracy for this set and confidence which is the average of individual credibility values in $D^{(\tau)}_{\text{pred}}$.

## 9 Reproduciblity

The code for generating the datasets of the clients, producing the figures as well as and running the different algorithms as well as computing the score functions for calibration is available at https://github.com/makni-mehdi/federated-swag in order to be able to reproduce all the experiments in this Bachelor Thesis Report.

## 10 Conclusion

The experiments conducted in this report have shown that one-shot methods have some limitations in the setting of Deep Learning. Indeed in that case, the loss-surface is highly non-convex and can depend on thousands of parameters. Besides, attempts to average weights did not behave well in Deep Learning and some papers have tried to overcome that issue. Furthermore, these experiments have shown that the models in Deep Learning are very sensitive to weight change compared to the Logistic Regression task. But taking advantage of that, we allowed the server to share the first layers of a neural network to the clients. This represents a function that allows the clients to preprocess their local datasets. Every client runs this method on the dataset to obtain a new preprocessed one. Now each client simply trains SWAG on Logistic Regression to account for uncertainty and share the parameters to the server who aggregates them within the context of the method FL-SWAG. We have shown that FL-SWAG achieves state-of-the-art results both in terms of accuracy as it usually outperforms every singular client's accuracy prediction and it performs well on calibration scores compared to SGLD and pSGLD even though they are not subject to Federated Learning constraints.
In short, this approach allows to have stable aggregation of weights in the server that not only respects the constraints of Federated Learning and privacy concerns but also accounts for calibration issues. Besides, we claim that if the server is allowed to share the first layers' weights of a neural network or even simply a blackbox function that helps clients preprocess their datasets before running Logistic Regression, we can achieve a considerable increase in performance. Indeed, state-of-the-art preprocessing allows the client to achieve state-of-the-art results. One final desirable property of this approach is that it is extremely efficient and only requires devices to train Logistic Regression locally which should be feasible in most mobiles in our the current era.

## References

Agarwal, A. and J. C. Duchi (2011). Distributed delayed stochastic optimization. *Advances in neural information processing systems 24*.

Agarwal, N., A. T. Suresh, F. X. X. Yu, S. Kumar, and B. McMahan (2018). cpsgd: Communication-efficient and differentially-private distributed sgd. *Advances in Neural Information Processing Systems 31*.

Amodei, D., C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané (2016). Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*.

Betancourt, M. (2017). A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*.

Brooks, S., A. Gelman, G. Jones, and X.-L. Meng (2011). *Handbook of markov chain monte carlo*. CRC press.

Brosse, N., C. Riquelme, A. Martin, S. Gelly, and É. Moulines (2020). On last-layer algorithms for classification: Decoupling representation from uncertainty estimation. *arXiv preprint arXiv:2001.08049*.

Chen, H.-Y. and W.-L. Chao (2021). Fedbe: Making bayesian model ensemble applicable to federated learning.

Deng, W., Y.-A. Ma, Z. Song, Q. Zhang, and G. Lin (2021). On convergence of federated averaging langevin dynamics. *arXiv preprint arXiv:2112.05120*.

Garipov, T., P. Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson (2018). Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems 31*.

Guha, N., A. Talwalkar, and V. Smith (2019). One-shot federated learning. *arXiv preprint arXiv:1902.11175*.

Guo, C., G. Pleiss, Y. Sun, and K. Q. Weinberger (2017). On calibration of modern neural networks. In *International Conference on Machine Learning*, pp. 1321–1330. PMLR.

Hard, A., K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage (2019). Federated learning for mobile keyboard prediction.

He, K., X. Zhang, S. Ren, and J. Sun (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Izmailov, P., W. J. Maddox, P. Kirichenko, T. Garipov, D. Vetrov, and A. G. Wilson (2020). Subspace inference for bayesian deep learning. In *Uncertainty in Artificial Intelligence*, pp. 1169–1179. PMLR.

Izmailov, P., D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson (2019). Averaging weights leads to wider optima and better generalization.

Izmailov, P., S. Vikram, M. D. Hoffman, and A. G. G. Wilson (2021). What are bayesian neural network posteriors really like? In *International Conference on Machine Learning*, pp. 4629–4640. PMLR.

Ji, Z., Z. C. Lipton, and C. Elkan (2014). Differential privacy and machine learning: a survey and review. *arXiv preprint arXiv:1412.7584*.

Kairouz, P., H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. (2019). Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*.

Kairouz, P., H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. (2021). Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning 14*(1–2), 1–210.

Kassab, R. and O. Simeone (2021). Federated generalized bayesian learning via distributed stein variational gradient descent.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report.

Langford, J., A. Smola, and M. Zinkevich (2009). Slow learners are fast. *arXiv preprint arXiv:0911.0491*.

Lecun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*(11), 2278–2324.

Li, C., C. Chen, D. Carlson, and L. Carin (2016). Preconditioned stochastic gradient langevin dynamics for deep neural networks. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Liu, D. and O. Simeone (2021). Channel-driven monte carlo sampling for bayesian distributed learning in wireless data centers.

Maddox, W. J., P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson (2019). A simple baseline for bayesian uncertainty in deep learning. *Advances in Neural Information Processing Systems 32*.

McMahan, H. B., E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas (2017). Communication-efficient learning of deep networks from decentralized data.

Minelli, M. (2018). *Fully homomorphic encryption for machine learning*. Ph. D. thesis, Université Paris sciences et lettres.

Neiswanger, W., C. Wang, and E. Xing (2013). Asymptotically exact, embarrassingly parallel mcmc. *arXiv preprint arXiv:1311.4780*.

Ramaswamy, S., R. Mathews, K. Rao, and F. Beaufays (2019). Federated learning for emoji prediction in a mobile keyboard.

van der Burg, S., L. Wiseman, and J. Krkeljas (2021). Trust in farm data sharing: reflections on the eu code of conduct for agricultural data sharing. *Ethics and Information Technology 23*(3), 185–198.

Verbraeken, J., M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer (2020). A survey on distributed machine learning. *ACM Computing Surveys (CSUR) 53*(2), 1–33.

Vono, M., V. Plassier, A. Durmus, A. Dieuleveut, and E. Moulines (2021). Qlsd: Quantised langevin stochastic dynamics for bayesian federated learning. *arXiv preprint arXiv:2106.00797*.

Wang, H., M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni (2020). Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*.

Wang, J., Z. Charles, Z. Xu, G. Joshi, H. B. McMahan, M. Al-Shedivat, G. Andrew, S. Avestimehr, K. Daly, D. Data, et al. (2021). A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*.

Welling, M. and Y. W. Teh (2011). Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688. Citeseer.

Wilson, A. G., P. Izmailov, M. D. Hoffman, Y. Gal, Y. Li, M. F. Pradier, S. Vikram, A. Foong, S. Lotfi, and S. Farquhar (2021). Evaluating approximate inference in bayesian deep learning.

Xiao, H., K. Rasul, and R. Vollgraf (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.