UNCERTAINTY QUANTIFICATION IN FEDERATED LEARNING

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

Mehdi Makni

7 avril 2022

Overview

Why do we need Calibration in Uncertainty Quantification?

- We can only interpret models' outputs as probabilities if the models are well calibrated.
- Crucial in Deep Learning to make reliable decisions.
- A robust probability estimation of events is critical in domains where human life or large sums of money are at stake.

What should be computed and how?

More concretely, we want to compute the posterior predictive distribution

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

This is usually obtained thanks to a Monte Carlo sampling procedure as follows:

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

We should be able to sample from the true posterior distribution.

- Example: HMC (Hamiltonian Monte Carlo): MCMC method which asymptotically samples from the true posterior.
- It is prohibitively expensive in Deep Learning.
- Serves as a ground-truth method but not practical for large-scale deployment.

Data in sensitive Domains

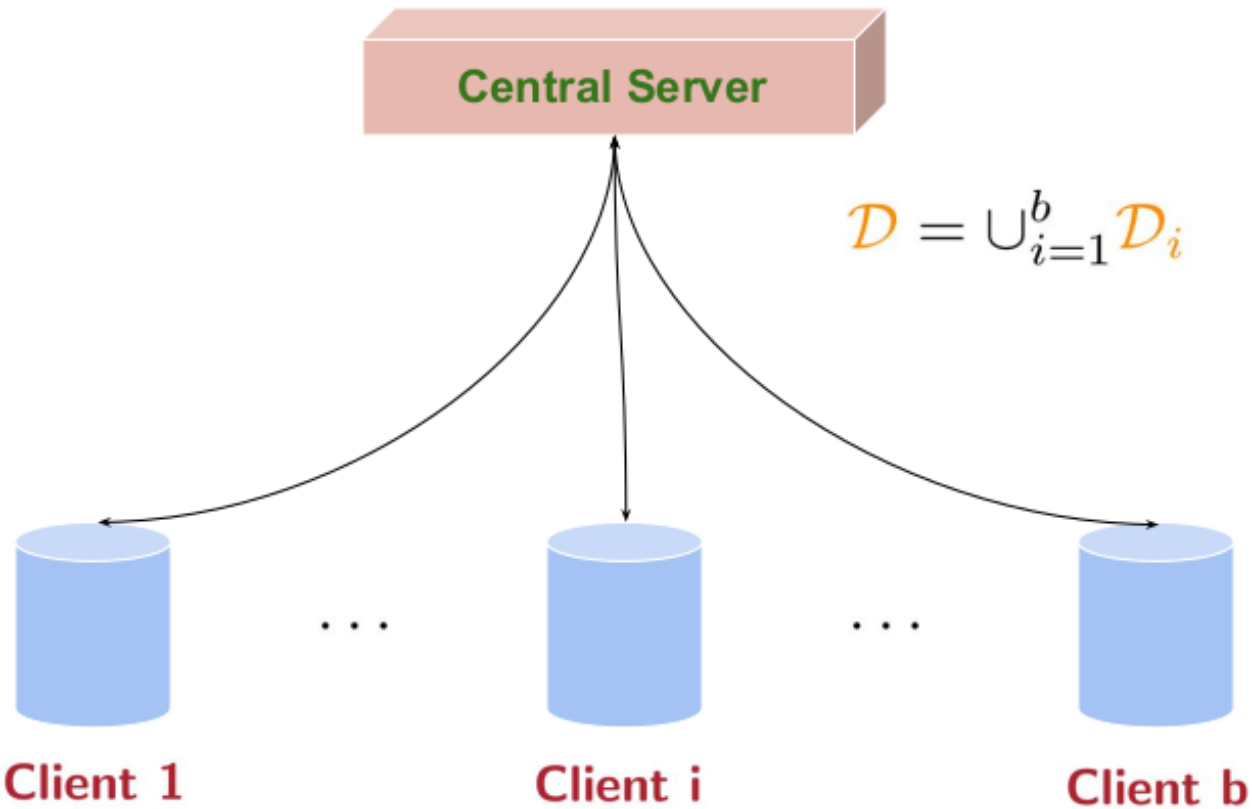
The domains where Uncertainty Quantification is demanded are characterized by the sensitivity of the data.

- Data Sharing which could be complicated due to government policies and privacy issues.
- Centralized storage represents a major security problem as it makes the database a SPOF (single point of failure) system.

$\Longrightarrow$ Federated Learning to the rescue !

Introduction

Federated Learning is a distributed approach for collaborative machine learning from decentralized data.



Constraints

The approach to Federated Learning carries multiple constraints:

- Non-IID (Heterogeneity)
- Imbalanced
- Limited Availability
- Computational Constraints
- Data Privacy constraints

Problem Formulation

Federated Learning Scenarios:

- Cross-device Federated Learning.
- Cross-silo Federated Learning (*).

Optimization Paradigms:

- Limiting the number of participating clients per round.
- Introducing Compression schemes during each Round.
- Reducing the number of Communication Rounds (*).

We consider the (*) setting and more precisely one-shot methods which communicate with the server only once.

SWAG is cool

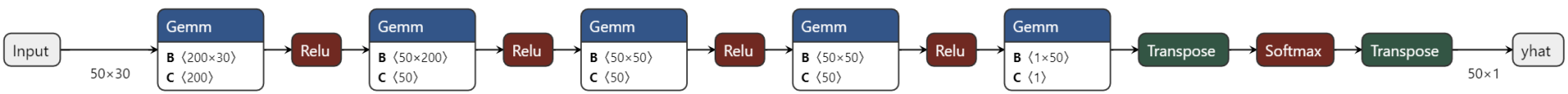
Some motivating facts to look for variants of SWAG:

- Some variants of SWAG won a NeurIPS (Leading conference in Machine Learning) competition.
- Intuitive: Tries to find a Gaussian in high dimensions to fit the weight space near the optimum.
- The covariance matrix is derived from the trajectory of SGD iterates near the averaged weight denoted θ_{SWA}.
- Efficient and performs well in Calibration Scores.

Visualizing SWAG thanks to PCA

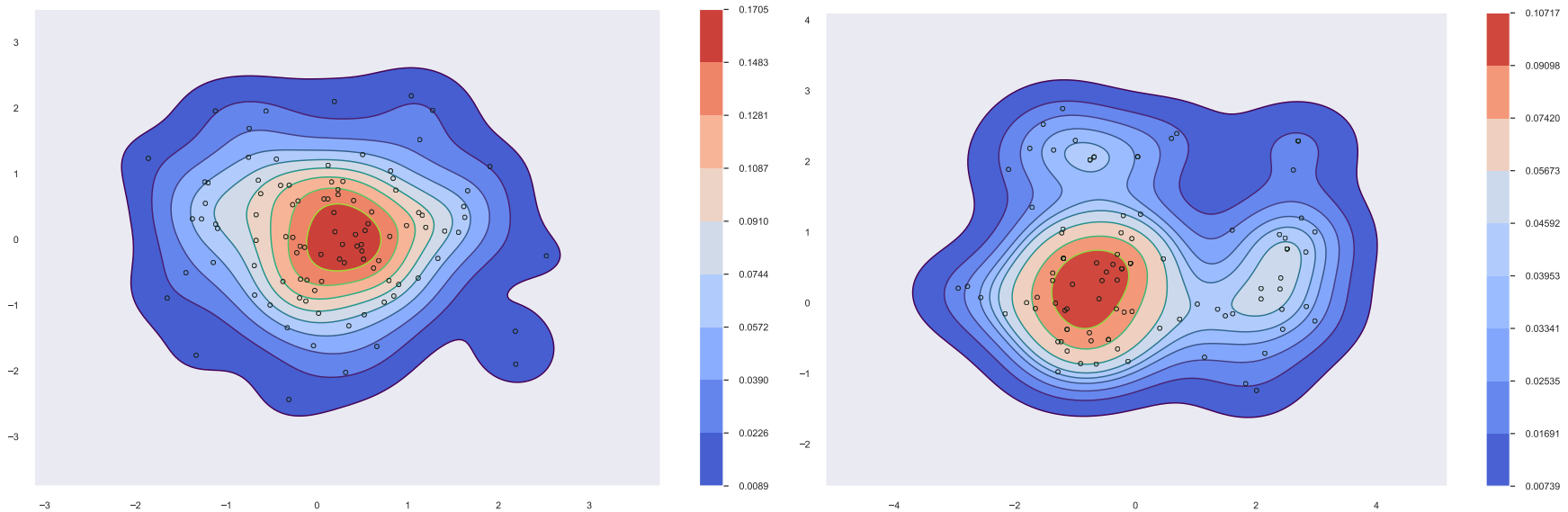
Dataset Used: Breast Cancer Wisconsin

Model Used: A simple feed-forward neural network with 3 hidden layers.



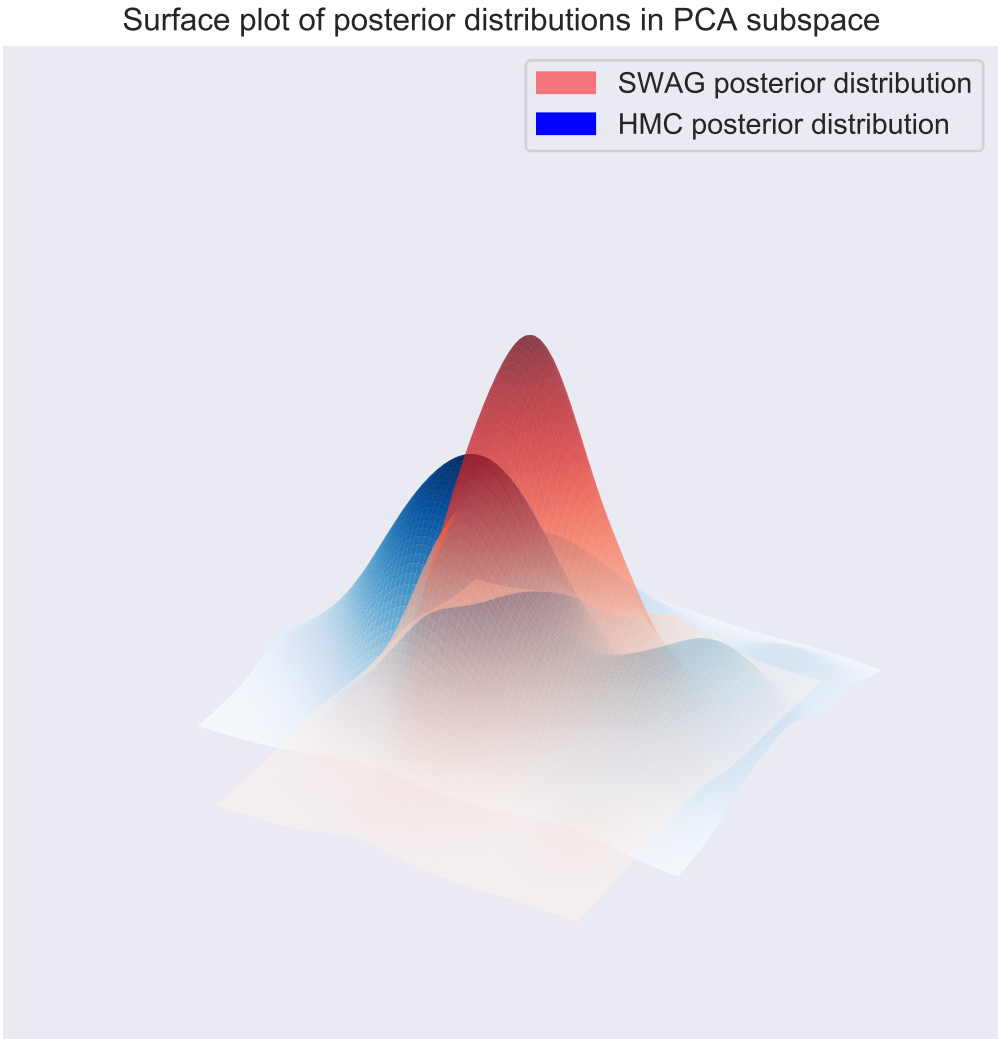
2D Projections

To the left, we have SWAG samples. To the right, we have HMC samples.



3D Visualization

This is the 3D visualization of the two distributions.



FL-SWAG is even cooler

In FL-SWAG, we train SWAG on the b client's local datasets. We then aggregate the results at the server as explained in this brief pseudocode:

Algorithm FL-SWAG

---

for $i = 1 \to b$ **do**

$\widehat{\mu}_i, \widehat{\Sigma}_i \leftarrow$ **Train**SWAG(client$_i$)

end for

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

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

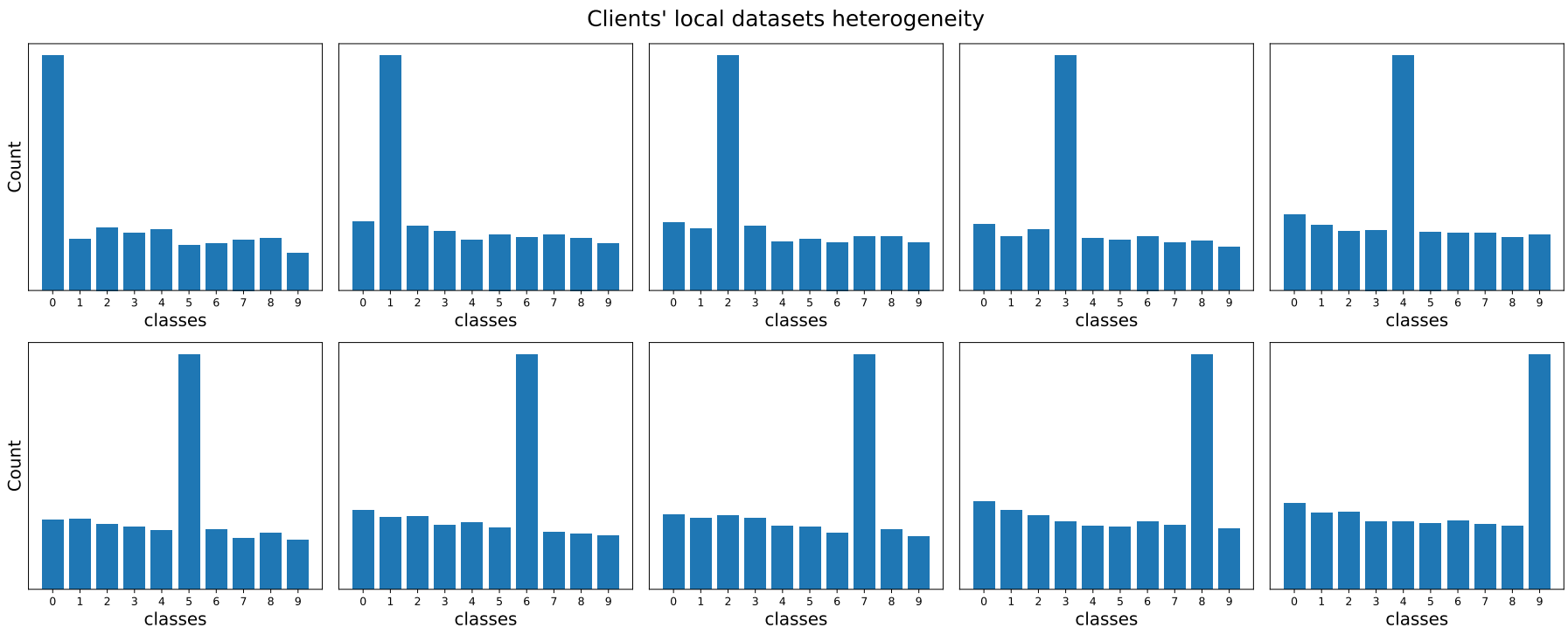
return μ_S, Σ_S

Motivation

- Efficient one-shot method: Training SWAG at the clients then one aggregation step.
- Suitable for online learning or when clients change their behavior (more concretely the parameters of the model at the clients' local datasets).
- Inspired from the formula of multiplying the distribution of Gaussians.
- Intuitive way to aggregate the parameters.

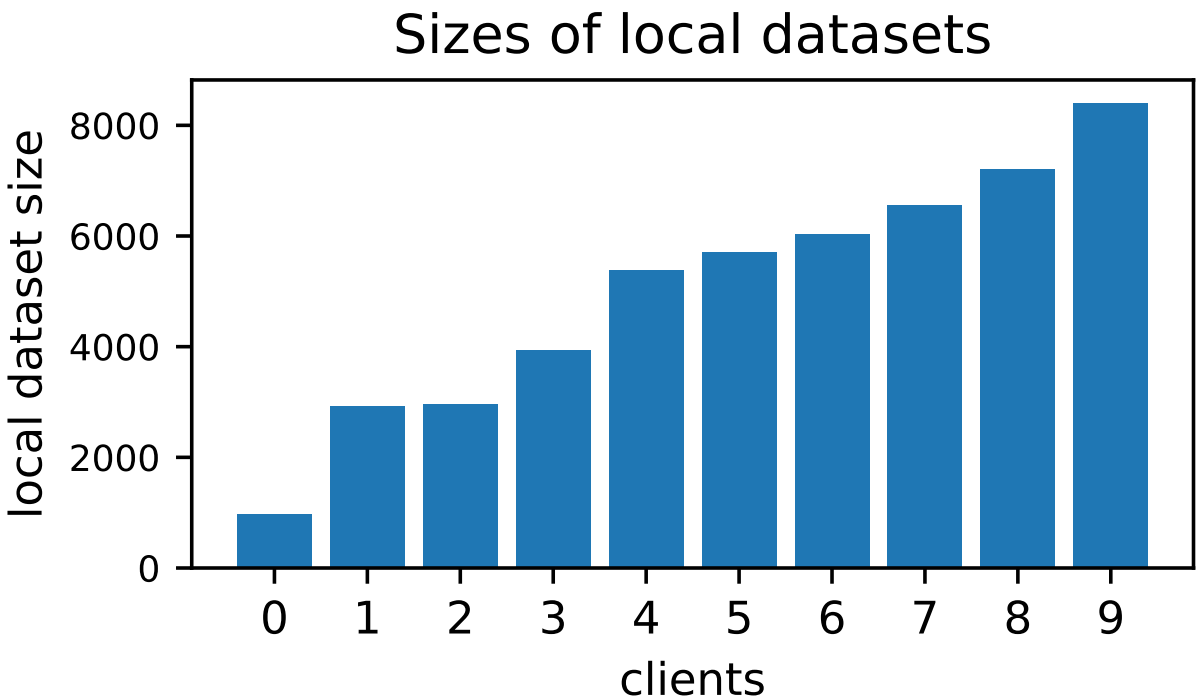
Data Splitting

We create a function that splits the data so that the local datasets across clients are heterogeneous and imbalanced. We obtain the following figures:



Data Splitting

We create a function that splits the data so that the local datasets across clients are heterogeneous and imbalanced. We obtain the following figures:



Methods for Comparison

In the experiments we will compare the performance of FL-SWAG against many methods.

In terms of Accuracy:

- FEDAVG Averaging the weights.
- WFEDAVG Mean of the weights multiplied by size of dataset.
- BAYAVG Averaging the predictions of each client. Not applicable in practise but strong benchmark.
- WBAYAVG Mean the predictions of each client multiplied by size of dataset.

In terms of Calibration Scores:

- SGLD Stochastic Gradient Langevin Dynamics.
- PSGLD preconditioned Stochastic Gradient Langevin Dynamics.

FL-SWAG with vanilla Logistic Regression

Datasets Used: MNIST - FashionMNIST

Model Used: Logistic Regression

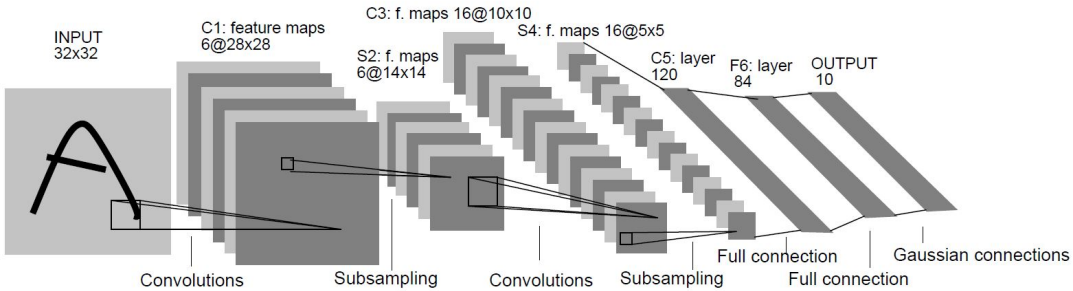
Weights	Accuracy (MNIST)	Accuracy (FashionMNIST)
client 1	84.07%	75.72%
client 2	87.41%	80.15%
client 3	87.75%	78.63%
client 4	89.57%	80.81%
client 5	90.21%	81.11%
client 6	90.09%	81.75%
client 7	90.77%	80.82%
client 8	90.29%	82.51%
client 9	90.69%	83.1%
client 10	90.8%	82.8%
FL-SWAG	90.96%	81.32%
FEDAVG	91.58%	83.0%
WFEDAVG	91.93%	83.63%
BAYAVG	91.65%	83.05%
WBAYAVG	91.98%	83.61%

FL-SWAG in Deep Learning

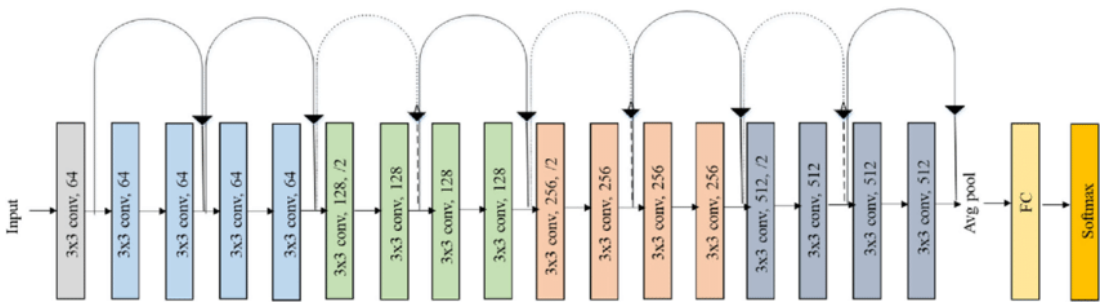
Datasets Used: MNIST - FashionMNIST

Models Used:

LeNet5



ResNet18



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).

Introducing Server preprocessing Step

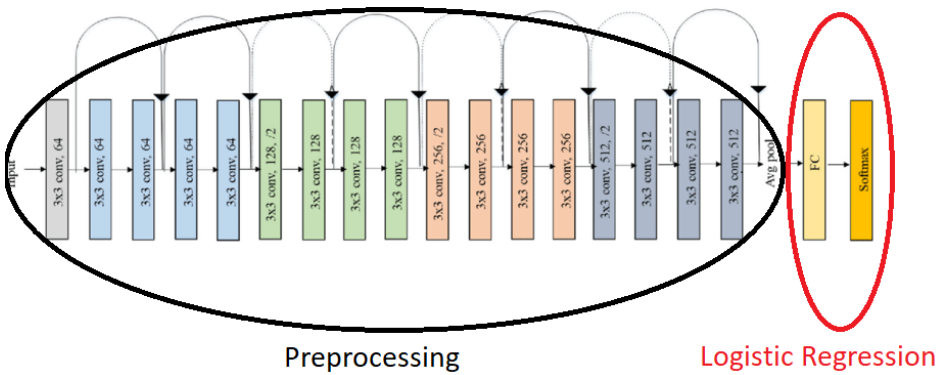
We allow clients to have extra shared information that does not impact the Federated Learning constraints.

In these approaches, we start by training a model at the level of the server. It is then allowed to share the initial weights to clients. We thus have these two variants.

- Clients start from a pretrained model then continue training on their local datasets.
- Clients treat their model as a highly-sophisticated preprocessing step followed by a Logistic Regression step and the initial layers of the model are freezed.

Why it might work?

- A neural network whose aim is classification commonly has for an output layer a Dense layer which will be treated as a Logistic Regression taking as input the result of the initial layers.



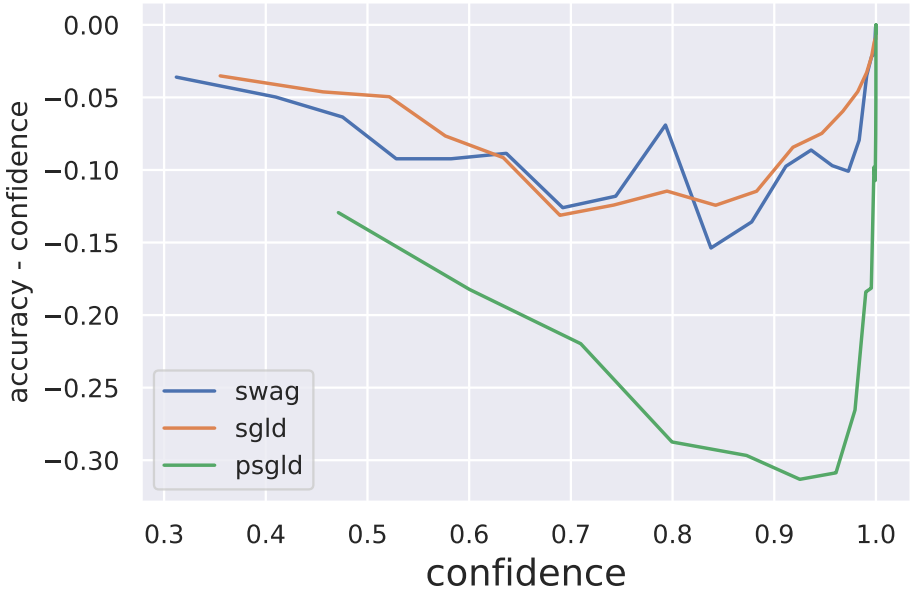
- It has been argued in the literature that there is limited value in adding multiple uncertainty layers.
- In this context, we compute the predictive posterior distribution as follows:

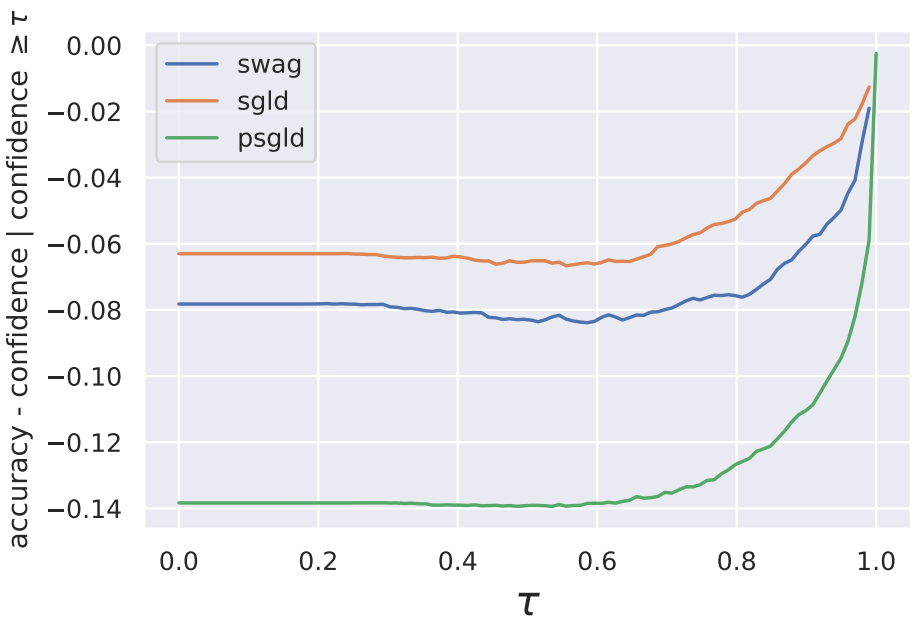
$$p(y \mid x, \mathcal{D}) = \int p(y|x, \theta_{\text{last\_layer}}) p(\theta_{\text{last\_layer}}|\mathcal{D})$$

Results

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

accuracy - confidence curves on CIFAR 10





Conclusion

- One-shot methods have limitations in the context of Deep Learning.
- Allowing the server to share extra information with the clients makes aggregation much more robust.
- Decomposing the network into preprocessing and Logistic Regression and Freezing the initial weights helped clients collaborate in learning a calibrated model that satisfies the Federated Learning constraints.
- State-of-the-art preprocessing gives state-of-the-art results.

Thank You !