entropy

Article

Probabilistic Predictions with Federated Learning

Adam Thor Thorgeirsson [1,2,*] and Frank Gauterin [2]

[1] Dr. Ing. h.c. F. Porsche AG, 71287 Weissach, Germany
[2] Institute of Vehicle System Technology, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany; frank.gauterin@kit.edu
* Correspondence: Adam.Thorgeirsson@partner.kit.edu

Abstract: Probabilistic predictions with machine learning are important in many applications. These are commonly done with Bayesian learning algorithms. However, Bayesian learning methods are computationally expensive in comparison with non-Bayesian methods. Furthermore, the data used to train these algorithms are often distributed over a large group of end devices. Federated learning can be applied in this setting in a communication-efficient and privacy-preserving manner but does not include predictive uncertainty. To represent predictive uncertainty in federated learning, our suggestion is to introduce uncertainty in the aggregation step of the algorithm by treating the set of local weights as a posterior distribution for the weights of the global model. We compare our approach to state-of-the-art Bayesian and non-Bayesian probabilistic learning algorithms. By applying proper scoring rules to evaluate the predictive distributions, we show that our approach can achieve similar performance as the benchmark would achieve in a non-distributed setting.

Keywords: probabilistic machine learning; federated learning; Bayesian deep learning; predictive uncertainty

check for updates

Citation: Thorgeirsson, A.T.; Gauterin, F. Probabilistic Predictions with Federated Learning. *Entropy* **2021**, *23*, 41. https://doi.org/10.3390/e23010041

Received: 20 November 2020
Accepted: 26 December 2020
Published: 30 December 2020

1. Introduction

Modern end devices generate large amounts of data, which enables the widespread, commercial application of machine learning (ML) algorithms. The data are distributed over a large group of end devices and are commonly transferred to a central server, where the learning of the ML models can be performed using the entire dataset. This poses two problems: transferring the data may lead to high communication costs and the privacy of the users may be compromised [1]. To counter these problems, ML can be performed on-device, so that the data are kept localized on the device and are not uploaded to a central server. The most prominent on-device ML methods are distributed learning [2,3], gossip learning [4] and federated learning [5]. In this work, we focus on the application of federated learning (FL) algorithms.

In FL, each end device learns from the local data, and a centralized server creates a global model by aggregating the model weights received from the devices at regular intervals. The global model is then sent back to the devices where the learning continues. FL is typically applied when a large dataset is desired, but sharing data between users is not possible or too expensive. In a distributed setting, data may not be independent and identically distributed (IID) and a robust model should take uncertainty into account. However, FL is not commonly applied to probabilistic models. In many applications, uncertainty of estimations or predictions can be significant. Bayesian deep learning (BDL) is commonly applied to account for uncertainty in neural networks (NNs) [6]. However, BDL methods are computationally expensive in comparison to non-Bayesian methods and hardware may as well be a limiting factor [7]. The inclusion of predictive uncertainty in distributed settings should therefore be addressed.

In this work, we apply FL to generate a probabilistic model. Inspired by related work on probabilistic predictions with NNs, we propose the learning of a probabilistic model through FL by introducing weight uncertainty in the aggregation step of the federated averaging (FedAvg) algorithm. In that way, the end devices can calculate probabilistic predictions but only have to learn conventional, deterministic models. This paper is organized as follows: In Section 2, we discuss probabilistic predictions with ML and give an overview of related work. In Section 3, we present our method, FedAvg-Gaussian (FedAG). In Section 4, we evaluate the performance of our method and compare the results to benchmarks from related literature. Finally, Section 5 gives concluding remarks and an outlook.

2. Related Work

A probabilistic prediction (or stochastic prediction) is when the prediction takes the form of a probability distribution, instead of a scalar value [8]. The application of ML in this topic is of significant relevance. Probabilistic predictions are commonly used in geology [9], electricity markets [10], urban water consumption [11], wind power [12], driver behavior [13], vehicle dynamics [14] and electric vehicle driving range applications [15]. Two prominent probabilistic prediction methods are BDL and ensemble methods, on both of which a summary was given by Ashuka et al. [16]. In BDL, the model parameters, e.g., weights $\mathbf{w}$, are random variables represented by probability distributions $p(\mathbf{w})$. With a dataset $\mathcal{D}$, consisting of features $\mathbf{x}$ and target variable y, the posterior distribution for the model weights can be derived using the Bayes' rule, which states that the posterior distribution is proportional to a prior probability $p(\mathbf{w}|\alpha)$ multiplied with likelihood $p(\mathcal{D}|\mathbf{w},\beta)$

$$p(\mathbf{w}|\mathcal{D},\alpha,\beta) \propto p(\mathbf{w}|\alpha)p(\mathcal{D}|\mathbf{w},\beta)\ , \tag{1}$$

where α is a precision parameter for the prior distributions on weights $\mathbf{w}$ and β is a noise precision parameter. For simplicity, we refer to the weight posterior distribution as $p(\mathbf{w}|\mathcal{D})$. To make predictions for new, unseen data, the predictive distribution is obtained with

$$p(y|\mathbf{x},\mathcal{D}) = \int p(y|\mathbf{x},\mathcal{D},\mathbf{w})p(\mathbf{w}|\mathcal{D})\mathrm{d}\mathbf{w}\ . \tag{2}$$

The exact computation of (1) and (2) is usually intractable due to the non-linearity of NNs [17]. The integration over the posterior is commonly approximated with Monte Carlo (MC) methods, such as Markov chain Monte Carlo (MCMC) or Hamiltonian Monte Carlo (HMC) [6]. Alternative approximation methods are extended variational inference [18] and cubature rules based on the unscented transformation [19].

A recent survey on BDL was given by Wang and Yeung [20]. Traditional Bayesian neural networks (BNNs) do not scale well and the posterior $p(\mathbf{w}|\mathcal{D})$ is usually difficult to calculate and sample from, but various approximation approaches have succeeded in creating probabilistic NNs. In variational inference (VI), the posterior $p(\mathbf{w}|\mathcal{D})$ is approximated with a well-defined distribution $Q(\mathbf{w}|\mathcal{D})$ and variational free energy $\mathcal{F}$ is minimized to minimize divergence between $p(\mathbf{w}|\mathcal{D})$ and $Q(\mathbf{w}|\mathcal{D})$ [21]. In Bayes by Backprop, the variational free energy is not minimized naïvely but approximately using gradient descent [22]. In probabilistic backpropagation (PBP), the posterior is determined with a calculation of a forward propagation of probabilities followed by a backwards calculation of gradients [23]. Gal and Ghahramani use dropout to achieve a mathematical equivalent of a Bayesian approximation without probabilistic weights [24]. Maddox et al. proposed SWA-Gaussian (SWAG), where an approximate posterior distribution over NN weights is determined by observing the stochastic gradient descent (SGD) trajectory during the learning process [25].

An established alternative to BDL is the use of ensembles to generate a probabilistic prediction. Therefore, multiple scalar predictions are combined to infer a probabilistic prediction. The predictions are either calculated with several different models or with a single model with varying initial conditions or input data. In a statistical post-processing of the ensemble predictions, a single probability density is derived [26]. A simple method is fitting a probability distribution to the predictions, e.g., a normal distribution $\mathcal{N}(\mu,\sigma^2)$, by setting μ equal to the ensemble mean and σ to the ensemble standard deviation [27]. Further techniques exist, such as the ensemble model output statistics (EMOS) method [28], which is common in the atmospheric sciences [29]. Numerical models, mechanistic models and ML algorithms can all be used as individual predictors in the ensemble, but in this work, we focus on the application of ML algorithms.

Deep ensembles are ensembles of NNs where each of the NNs predicts the parameters of a predictive distribution, e.g., μ and σ, and the ensemble prediction is then a mixture of Gaussians [7]. Snapshot ensembles are generated by taking snapshots of NN weights at local minima during the training process, thus achieving an ensemble of NNs by training a single NN [30]. Fast geometric ensembling also trains a single NN and explores the weight space to find a set of diverse weight samples with minimal loss, thereby generating an ensemble of NNs [31]. Depth uncertainty networks are ensembles of sub-networks of increasing depth which share weights, thus needing only a single forward pass [32]. Ensembles of other ML algorithms also exist, e.g., gradient boosting (GB) ensembles [33]. Out of these ensemble methods, deep ensembles (DE) have recently shown the quite promising results in terms of prediction performance. The nature of DE has a certain resemblance to distributed methods, i.e., independent and parallel training of multiple NNs.

The learning of probabilistic ML models in a distributed and federated setting is the central challenge of our work. In partitioned variational inference (PVI), federated approximate learning of BNNs is presented [34]. Sharma et al. presented an extension of PVI including differential privacy [35]. However, probabilistic predictions of continuous variables are not implemented and we are therefore unable to use these methods as benchmarks. Concurrent to our work, several articles on probabilistic FL were published. Kassab and Simeone introduced distributed Stein variational gradient descent (DSVGD), where non-random and interacting particles represent the model global posterior. Iterative updates of the particles are performed on the devices by minimizing the global free energy [36]. Al-Shedivat et al. proposed federated posterior averaging (FedPA), where the devices use MCMC to infer approximations of the local posteriors, and the server computes an estimate of the model global posterior [37]. Zhang et al. used FedAvg with differential privacy to learn a Bayesian long short-term memory (LSTM) network, where Monte Carlo dropout is applied to compute probabilistic forecasts of solar irradiation [38]. In the next section, we propose our alternative method for the application of FL to probabilistic ML models.

3. Federated Learning with Predictive Uncertainty

Our proposed method, FedAvg-Gaussian (FedAG), builds on the federated averaging (FedAvg) algorithm [5]. In FedAvg, clients perform quick local updates on the weights, which are then aggregated in a central server. In turn, the aggregated weights are then returned to the clients for further learning. FedAvg does not consider predictive uncertainty. However, before the weights are aggregated, information on their distribution over the clients is known. Xiao et al. showed that during FL, client weights become increasingly correlated but not closer to each other in terms of distance metrics [39]. This fact may be a sign that the client weights are a good, approximate Bayesian marginalization, i.e., the weigths represent multiple *basins of attraction* in the posterior [40]. In our algorithm, this information is used to introduce weight uncertainty in the aggregation step of the FedAvg algorithm. Therefore, a probabilistic model is approximated by treating the set of local weights in the ensemble as an empirical posterior distribution for the weights of the global model. Using the probabilistic model, inference is performed by calculating predictive distributions for new, unseen data.

A pseudo-code for FedAG is shown in Algorithm 1. In the aggregation step, a probability distribution is fitted to the set of client weights. The choice of this distribution is arbitrary, but for simplicity, we consider normal distributions in this work. Hence, the posterior distributions are found by calculating the mean value $\mu_{\mathbf{w}}$ and variance $\sigma^2_{\mathbf{w}}$ of weights $\mathbf{w}^{(k)}$. In turn, the posterior distributions $p(\mathbf{w}|\mathcal{D})$ are returned to the clients. The clients use the expected value, i.e., the mean value $\mu_{\mathbf{w}}$ of the weight posterior distributions to further iterate local updates to the global model using their own data, but calculate probabilistic predictions with $p(\mathbf{w}|\mathcal{D})$.

Algorithm 1 FedAvg-Gaussian (FedAG). C is the fraction of devices used in each round, K is the total number of devices, $\mathcal{D}_k$ is the data observed by device k, B is the batch size, E is the number of local epochs, η is the learning rate and l is the squared loss function.

```
Server executes:
initialize $\mathbf{w}_0$
for each round $t = 1, 2, ...$ do
    $m \leftarrow \max(C \cdot K, 1)$
    $S_t \leftarrow$ (random set of $m$ clients)
    for each client $k \in S_t$ do
        $\mathbf{w}_{t+1}^{(k)} \leftarrow \text{ClientUpdate}(k, p(\mathbf{w}_t|\mathcal{D}))$
    end
    $p(\mathbf{w}_{t+1}|\mathcal{D}) \leftarrow \mathcal{N}(\mu_{\mathcal{D}}(\mathbf{w}_{t+1}^{(k)}), \sigma_{\mathcal{D}}^2(\mathbf{w}_{t+1}^{(k)}))$
    return $p(\mathbf{w}_{t+1}|\mathcal{D})$ to clients
end

ClientUpdate$(k, p(\mathbf{w}_t|\mathcal{D}))$: // Run on client k
$\mathcal{B} \leftarrow$ (split $\mathcal{D}_k$ into batches of size $B$)
$\mathbf{w} \leftarrow \mathbb{E}(p(\mathbf{w}_t|\mathcal{D}))$
for each local epoch $i = 1$ to $E$ do
    for batch $b \in \mathcal{B}$ do
        $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla \ell(\mathbf{w}; b)$
    end
end
return $\mathbf{w}$ to server
```

As in FedAvg, the clients minimize the mean squared error (MSE). The client updates are therefore fast and do not require extensions in order to learn a probabilistic model. FedAG is therefore significantly less complicated than PVI, DSVGD and FedPA, which is beneficial when resources such as computing performance and storage are limited. Additionally, the only target variable during training is μ, so that the amount of operations is smaller in comparison to DE. Figure 1 shows an overview of the training process where a network of end devices learns a probabilistic model. For w_i, the clients return their local updates, to which a normal distribution is fitted to generate a posterior probability distribution $p(w_i|\mathcal{D})$. The distributions $p(w_i|\mathcal{D})$ constitute the weights of the NN with input variable x, hidden units H_i, bias I and target variable $\hat{y}$. FedAG does not require prior probabilities on the weights.

As mentioned in Section 2, an exact calculation of the integral in (2) for the predictive distribution is generally intractable and some approximation is needed. We propose two variations for our algorithm: ordinary Monte Carlo (OMC) and non-parametric bootstrapping. In OMC, M sets of the weights are drawn from the posterior distributions to calculate M scalar predictions $\hat{y}_k$ for the target variable y [41]. It may seem strange to draw sets of sample weights from a distribution created by aggregating sets of sample weights. An alternative would be to use the sample weights from the clients directly to calculate the predictions. In a sense, this resembles non-parametric bootstrapping to create an ensemble [42]. In that way, $\hat{y}_k$ are calculated directly from the client updates. In this work, we will use this latter sampling method. The predictive distribution is approximated with a normal distribution $\mathcal{N}(\hat{\mu}_y, \hat{\sigma}_y^2)$:

$$p(y|\mathbf{x}, \mathcal{D}) = \int p(y|\mathbf{x}, \mathcal{D}, \mathbf{w}) p(\mathbf{w}|\mathcal{D}) \mathrm{d}\mathbf{w} := \mathcal{N}(\hat{\mu}_y, \hat{\sigma}_y^2) \tag{3}$$

$$\hat{\mu}_y \approx \frac{1}{M} \sum_{k=1}^{M} \hat{y}(\mathbf{x}, \mathbf{w}^{(k)}) \tag{4}$$

$$\hat{\sigma}_y^2 \approx \left(\frac{1}{M} \sum_{k=1}^{M} \left[\hat{y}(\mathbf{x}, \mathbf{w}^{(k)}) \right]^2 \right) - \hat{\mu}_y^2 , \tag{5}$$

where $\hat{y}(\mathbf{x}, \mathbf{w}^{(k)})$ is a prediction calculated with features $\mathbf{x}$ and weights $\mathbf{w}^{(k)}$. In the case of a linear model, the predictive distribution takes the form

$$p(y|\mathbf{x}, \mathcal{D}, \alpha) = \mathcal{N}\left(\mu_{\mathbf{w}}^{\mathrm{T}} \mathbf{x}, \beta^{-1} + \mathbf{x} \left[\sigma_{\mathbf{w}}^2 \mathbf{I} \right] \mathbf{x}^{\mathrm{T}} \right) , \tag{6}$$

where β is a noise precision parameter for data $\mathcal{D}$ and is considered to be independent of the distribution of the weights $\mathbf{w}$, $\mathbf{I}$ is the identity matrix, $\mu_{\mathbf{w}}$ and $\sigma_{\mathbf{w}}^2$ are the mean and variance of the weight posterior distribution $p(\mathbf{w}|\mathcal{D})$ [17].

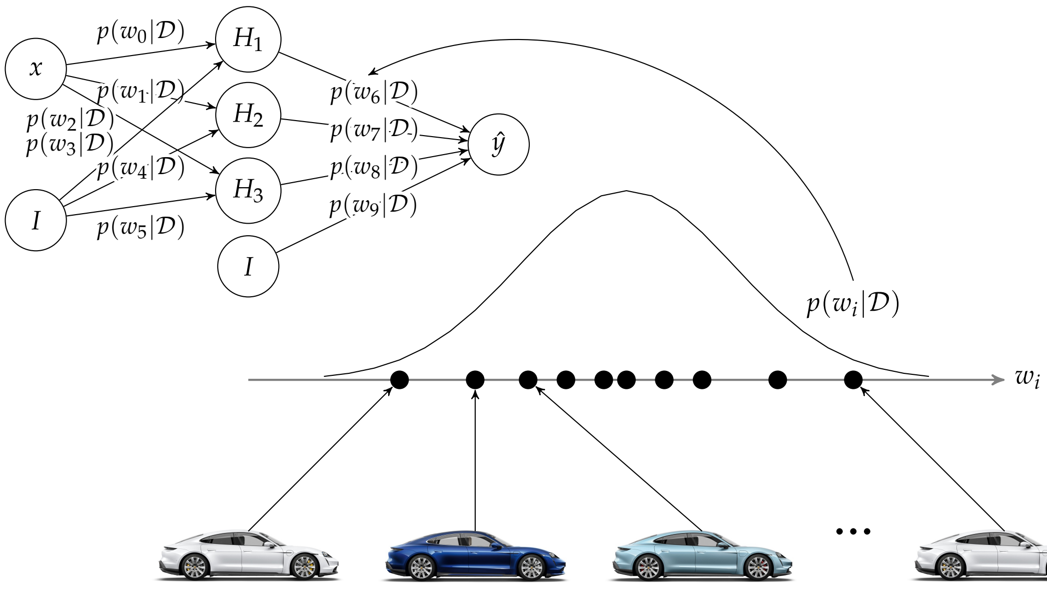


Figure 1. A network of end devices learns a probabilistic model.

4. Experimental Evaluation

As commonly done in the field of ML, we validate our proposed method with empirical data. In this section, we describe our experiments and analyze the results. In Section 4.1, we present a summary of proper scoring rules, which are necessary for the evaluation of probabilistic predictions. In Section 4.2, we apply FedAG to toy regression data. Section 4.3 shows the setup of the empirical validations and in Section 4.4, we present the results.

4.1. Proper Scoring Rules

To appropriately evaluate probabilistic predictions, proper scoring rules are needed. A scoring rule S is proper if

$$\mathbb{E}_{y \sim P}[S(P, y)] \geq \mathbb{E}_{y \sim P}[S(Q, y)] , \tag{7}$$

where P is the true distribution of outcomes y and Q is the predictive distribution or any other probability distribution [43]. The scoring rule is strictly proper if the equality holds only when $P = Q$. Popular scoring rules for the prediction of continuous variables are the logarithmic score $\mathcal{L}(F, y)$, the continuous ranked probability score (CRPS) and its generalization, the energy score $\mathrm{ES}(F, y)$ [44]. In related work, negative log-likelihood (NLL) has been favored as a performance indicator. NLL is equal to the negative logarithmic score and is therefore also a proper scoring rule. Furthermore, NLL is unitless which is advantageous when evaluating a model's performance on different datasets.

A good prediction is well calibrated and sharp. Calibration is the statistical consistency between the predictive distribution and the observation of the target variable. Sharpness measures the concentration of the predictive distribution. NLL measures both calibration and sharpness whereas root mean square error (RMSE) only measures calibration. Separate measures for calibration and sharpness allow a more detailed comparison. The width of a central prediction interval, e.g., 50%, was suggested by Gneiting and Raftery as a measure for sharpness [44]. As all candidate algorithms in this work calculate a prediction in the form of a normal distribution, the standard deviation appropriately measures the sharpness by indicating the width of the central 68% prediction interval. This is also called determinant sharpness (DS):

$$\mathrm{DS} = \det(\Sigma)^{1/2d}\,, \tag{8}$$

where $\Sigma \in \mathbf{R}^{d \times d}$ is the covariance matrix of the predictive distribution and d is the dimension of the target variable. In our evaluation, we use the proper scoring rule NLL, as well as RMSE and DS.

4.2. Regression with Toy Data

To analyze the performance of our method on simple data, we generate a one-dimensional toy dataset as suggested by Hernández-Lobato and Adams [23]. In our analysis, 10 workers draw 16 independent examples from $y = x^3 + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 3^2)$. Each worker trains a NN with a single hidden layer with 100 hidden units from these data according to Algorithm 1. The data are sampled in the interval $[-4, 4]$ but predictions are calculated for the interval $[-6, 6]$. Figure 2 shows the resulting probabilistic predictions after 1, 3 and 5 communication rounds. The results after $t = 5$ rounds show that FedAG can calculate accurate probabilistic predictions with low but appropriate uncertainty for input data close to the observed training data. For input data farther away from observed data, the uncertainty is high. The prediction interval thus includes the ground truth, despite the scarce training data, making the prediction superior to those calculated after rounds $t = 1$ and $t = 3$. The predictions after $t = 5$ rounds are similar to those reported by [7,23].

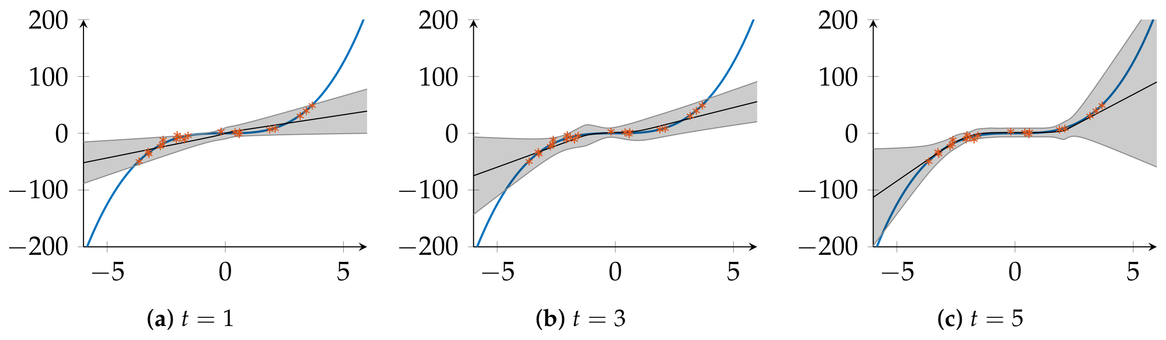


Figure 2. Results on regression for toy data after 1, 3 and 5 rounds. The blue line represents the ground truth, the orange points are exemplary observed noisy training data, the black line is the mean value of the predictive distribution and the grey area demarcates a prediction interval containing ± 3 standard deviations.

4.3. Experiment Setup

For the empirical validation, we implemented our method with two models: a NN with a single hidden layer and a linear regression model, denoted FedAG$_{N=1}$ and FedAG$_{N=0}$, respectively, where N represents the number of hidden layers. We use the experiment setup described by [23], which was also used by [7,24]. There, 10 datasets from the UCI Machine Learning Repository are used [45]. Table 1 shows a summary of the corresponding datasets. To ensure fair comparability to benchmark algorithms, the standard datasets are used.

Table 1. Summary of the UCI datasets for regression.

Datasets	Observations	Features
Boston Housing	506	13
Concrete Compression Strength	1030	8
Energy Efficiency	768	8
Kin8nm	8192	8
Naval Propulsion Power Plant	11,934	16
Combined Cycle Power Plant	9568	4
Protein Structure	45,730	9
Red Wine Quality	1599	11
Yacht Hydrodynamics	308	6
Year Prediction MSD	515,345	90

We compare the performance of FedAG to three benchmarks: Bayesian linear regression (BLR) [17], variational inference (VI) [21], and deep ensembles (DE) [7], all of which are implemented in a non-distributed setting. We use Gaussian posteriors in VI and the DE consists of 5 networks. The NNs trained using VI, DE, and FedAG all have the same architecture with 50 hidden units with rectified linear unit (ReLU) activation functions in a single hidden layer. For the *Protein Structure* and *Year Prediction MSD* datasets, 100 hidden units are used. A 20-fold cross validation is performed to evaluate test performance, where $E = 40$ passes over the available training data are done. For the *Protein Structure* dataset, a 5-fold cross validation is performed and for the *Year Prediction MSD* dataset, the specified split is used. The linear models, FedAG$_{N=0}$ and BLR, are validated in the same manner. In the federated setting, $K = 10$ devices are simulated with $C = 1$ and batch size $B = 1$. K and C are chosen so that the amount of data per device is maximized, subject to the condition that the number of devices is sufficiently large to enable an accurate approximation of the posterior distribution in the aggregation step. The training data are randomly divided into K equally large shards, each of which is assigned to a simulated device. Hence, each observation is uniquely assigned to one device.

The training of a linear model is a convex optimization and we expect that FedAG$_{N=0}$ should need no more than $t = 1$ rounds to converge. On the contrary, the training of a NN is usually a non-convex optimization and $t > 1$ rounds are therefore required for convergence of FedAG$_{N=1}$ in our setting. When each device has a limited amount of data, such as in small datasets or when the number of devices is increased, an even higher number of communication rounds might be required. Strong baselines in BDL are important and we try to generate a fair basis for the comparison of FedAG and the benchmarks [46]. For BLR and FedAG$_{N=0}$, appropriate precision parameters for the variance of the target variable are estimated using the variance of the training data. In addition, conjugate priors given by unit Gaussians are used for the weight posterior distributions $p(\mathbf{w}|\mathcal{D})$ in BLR.

4.4. Results

With the experiment setup and proper scoring rules, we can evaluate the performance of FedAG and the benchmarks. In the following, we present the results of the validation. Table 2 shows the mean NLL and standard error for the algorithms on all dataset and Table 3 shows the RMSE and standard error. The results for VI and DE are reported by [7,23], respectively. The entries in bold denote the best performing model(s), where the performance is considered similar if the standard error intervals overlap.

Table 2. Mean NLL and standard error of the predictions. The entries in bold denote the best performing model(s).

Dataset	**BLR**	**VI**	$\mathbf{FedAG}_{N=0}^{(t=1)}$	$\mathbf{FedAG}_{N=1}^{(t=5)}$	**DE**
Boston	3.07 ± 0.03	2.90 ± 0.07	3.02 ± 0.03	$\mathbf{2.58 \pm 0.06}$	$\mathbf{2.41 \pm 0.25}$
Concrete	3.78 ± 0.02	3.39 ± 0.02	3.76 ± 0.03	$\mathbf{3.21 \pm 0.04}$	$\mathbf{3.06 \pm 0.18}$
Energy	5.12 ± 0.05	2.39 ± 0.03	5.31 ± 0.06	2.07 ± 0.04	$\mathbf{1.38 \pm 0.22}$
Kin8nm	1.17 ± 0.04	-0.90 ± 0.01	1.03 ± 0.04	-0.87 ± 0.01	$\mathbf{-1.20 \pm 0.02}$
Naval Propulsion	-3.55 ± 0.02	-3.73 ± 0.12	-3.45 ± 0.01	-3.21 ± 0.01	$\mathbf{-5.63 \pm 0.05}$
Power Plant	2.97 ± 0.01	2.89 ± 0.01	2.94 ± 0.01	2.92 ± 0.01	$\mathbf{2.79 \pm 0.04}$
Protein	3.07 ± 0.00	2.99 ± 0.01	3.08 ± 0.00	2.95 ± 0.00	$\mathbf{2.83 \pm 0.04}$
Red Wine	1.50 ± 0.07	$\mathbf{0.98 \pm 0.01}$	$\mathbf{1.01 \pm 0.03}$	$\mathbf{0.99 \pm 0.02}$	$\mathbf{0.94 \pm 0.12}$
Yacht	3.63 ± 0.05	3.44 ± 0.16	4.02 ± 0.07	1.92 ± 0.06	$\mathbf{1.18 \pm 0.21}$
Year Prediction	3.73± NA	3.86± NA	3.72± NA	3.66± NA	**3.35± NA**

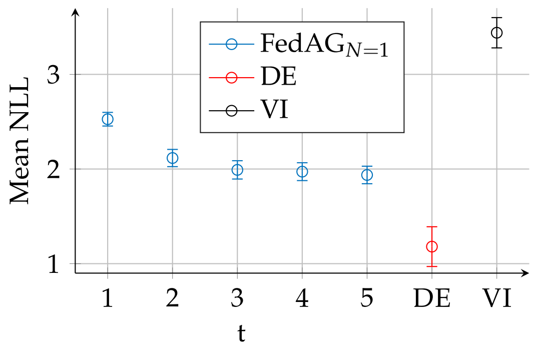
Table 3. RMSE and standard error of the predictions. The entries in bold denote the best performing model(s).

Dataset	**BLR**	**VI**	$\mathbf{FedAG}_{N=0}^{(t=1)}$	$\mathbf{FedAG}_{N=1}^{(t=5)}$	**DE**
Boston	4.87 ± 0.22	$\mathbf{4.32 \pm 0.29}$	4.96 ± 0.22	$\mathbf{4.07 \pm 0.18}$	$\mathbf{3.28 \pm 1.00}$
Concrete	10.58 ± 0.33	7.13 ± 0.12	10.52 ± 0.33	$\mathbf{6.50 \pm 0.20}$	$\mathbf{6.03 \pm 0.58}$
Energy	4.35 ± 0.14	2.65 ± 0.08	4.36 ± 0.14	$\mathbf{2.02 \pm 0.07}$	$\mathbf{2.09 \pm 0.29}$
Kin8nm	0.20 ± 0.00	0.10 ± 0.00	0.20 ± 0.00	0.10 ± 0.00	$\mathbf{0.09 \pm 0.00}$
Naval Propulsion	0.01 ± 0.00	$\mathbf{0.00 \pm 0.00}$	0.01 ± 0.00	0.01 ± 0.00	$\mathbf{0.00 \pm 0.00}$
Power Plant	4.74 ± 0.05	4.33 ± 0.04	4.56 ± 0.05	4.45 ± 0.05	$\mathbf{4.11 \pm 0.17}$
Protein	5.18 ± 0.02	4.84 ± 0.03	5.18 ± 0.02	$\mathbf{4.63 \pm 0.02}$	$\mathbf{4.71 \pm 0.06}$
Red Wine	$\mathbf{0.65 \pm 0.02}$	$\mathbf{0.65 \pm 0.01}$	$\mathbf{0.65 \pm 0.02}$	$\mathbf{0.65 \pm 0.02}$	$\mathbf{0.64 \pm 0.04}$
Yacht	9.12 ± 0.52	6.89 ± 0.67	9.12 ± 0.52	2.29 ± 0.15	$\mathbf{1.58 \pm 0.48}$
Year Prediction	9.51± NA	9.03± NA	9.51± NA	9.35± NA	**8.89± NA**

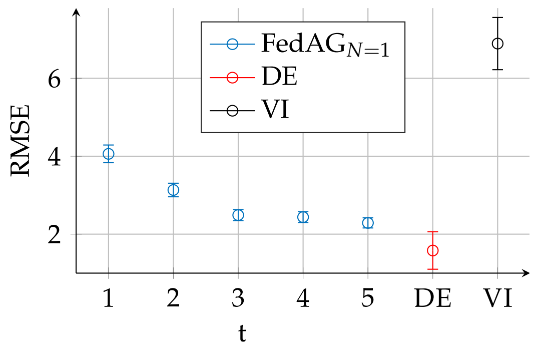
The performance of the two linear models, BLR and FedAG$_{N=0}$ is similar. In 8 out of 10 datasets, FedAG$_{N=0}^{(t=1)}$ performs similarly or slightly better than BLR in terms of NLL. BLR significantly outperforms FedAG$_{N=0}^{(t=1)}$ only in two datasets, the *Energy Efficiency* and *Yacht Hydrodynamics* datasets, which are also two of the smallest datasets. Hence, each worker only has access to a small amount of data. In 9 out of 10 datasets, the performance of FedAG$_{N=0}^{(t=1)}$ and BLR is almost identical in terms of RMSE.

In the results for NNs, the difference between the three algorithms, VI, FedAG$_{N=1}$ and DE is somewhat significant. DE achieve the best results, followed by FedAG$_{N=1}^{(t=5)}$ and VI. In 8 out of 10 datasets, FedAG$_{N=1}^{(t=5)}$ outperforms VI in terms of NLL and in 3 out of 10 datasets, the performance of FedAG$_{N=1}^{(t=5)}$ approaches that of DE. In terms of RMSE, the performance of FedAG$_{N=1}$ and DE is similar in 5 out of 10 datasets. Further rounds ($t > 5$) do not improve the results of FedAG$_{N=1}$ significantly.

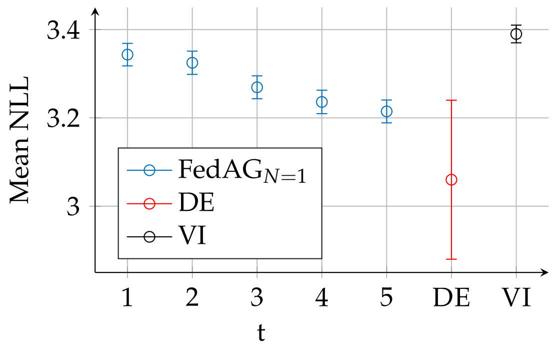
To further compare the performance of VI, DE and FedAG$_{N=1}$ over the course of the communication rounds, we look at the dataset *Concrete Compression Strength*, where the performance of the methods is similar, and the dataset *Yacht Hydrodynamics* where DE show a significant advantage in terms of NLL. Figure 3 shows the prediction performance (NLL and RMSE) of the algorithms on these two datasets. In Figure 3a,b, FedAG$_{N=1}$ outperforms VI already after $t = 1$ rounds, both in terms of NLL and RMSE. However, FedAG$_{N=1}$ reaches a certain saturation and cannot match the performance of DE, despite a significant improvement in NLL and RMSE after $t = 5$ rounds. In Figure 3c,d, FedAG$_{N=1}$ and VI show similar performance after $t = 1$ rounds. With increasing number of communication rounds t, NLL and RMSE of FedAG improve. After $t = 5$ rounds, the results of FedAG$_{N=1}$ and DE overlap, i.e., the algorithms achieve similar performance, though DE still retains a slight advantage. In the initial round of FedAG, different devices might find weights corresponding to different minima of the NN's loss function, so that the global model's initial weight posterior distributions are not optimal. The loss function of a NN with ReLU activation functions can be expressed as a polynomial function of the weights in the network, whose degree is the number of layers, and whose number of monomials is the number of paths from inputs to output [47]. We can therefore expect that loss functions of small NNs have few local minima, and that the global minimum can be found within relatively few communication rounds in FedAG. Accordingly, larger NNs might require more communication rounds. On the two datasets in Figure 3, we observe how the performance of FedAG$_{N=1}$ gradually improves with increased rounds t. This can also be observed on other datasets.



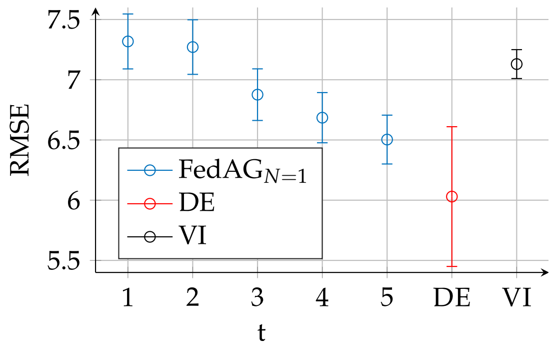
a) Yacht Hydrodynamics NLL



b) Yacht Hydrodynamics RMSE



c) Concrete Compression Strength NLL



d) Concrete Compression Strength RMSE

Figure 3. Prediction performance of FedAG$_{N=1}$, DE and VI in terms of NLL and RMSE on the datasets *Yacht Hydrodynamics* and *Concrete Compression Strength*.

Another important property of the predictive distributions is their sharpness, which we measure with determinant sharpness (DS). In Table 4, the mean DS of the predictive distributions calculated with FedAG$_{N=1}^{(t=5)}$ and DE are shown. Of the datasets that exhibit similar performance in terms of NLL, *Boston Housing* and *Concrete Compression Strength* can be predicted with greater sharpness by FedAG than DE, whereas DE's predictions of *Red Wine Quality* are sharper on average. In 7 out of 10 datasets, FedAG predicts on average a sharper distribution than DE.

4.5. Computational and Communication Complexity

In addition to the predictive performance of the algorithms, their computational and communication complexity is of significant importance. The candidate algorithms have different computational complexity at training time and at testing time. The two linear models, BLR and FedAG$_{N=0}$, have the same structure and the same amount of parameters. The predictive distributions can be computed analytically with (6) and no sampling is required. The NNs trained using VI, DE, and FedAG are, on the other hand, more complex. VI and FedAG learn probabilistic NNs with Gaussian posterior, whereas DE are ensembles consisting of deterministic NNs with scalar weights, but with two output variables. VI maximizes a lower bound on the marginal likelihood of the NN. First, a Monte Carlo approximation for the lower bound is computed, which is then optimized using SGD. The computational complexity at training time is therefore higher in VI than in DE and FedAG, where SGD is applied directly. VI and FedAG approximate predictive distributions using Monte Carlo sampling from the posterior distributions. Contrarily, DE only have to analytically compute the two output variables of the 5 networks in the ensemble. Subsequently, the predictive distribution is approximated as a mixture of the individually computed normal distributions. The computational complexity at testing time is therefore higher in VI and FedAG than in DE.

The communication complexity of FedAG is different from that of FedAvg. FedAG learns a posterior distribution for each weight of the model. Therefore, its communication complexity is somewhat higher than that of FedAvg. If a Gaussian posterior is assumed, each distribution is defined by its mean and standard deviation. Compared to FedAvg, the global model has twice the amount of parameters. A single parameter can be assumed to be a 32 bit floating-point value. For the NNs considered in this work (single hidden layer, 6–90 features, 50 or 100 hidden units), the total data size is in the range from 1604 B to 36,804 B in FedAvg and from 3208 B to 73,608 B in FedAG. The communication complexity of sending the global model to the clients in FedAG can be up to two times higher than in FedAvg, depending on the communication overhead. However, the client updates only include scalar weights $\mathbf{w}$, so the upload communication complexity in FedAG is the same as in FedAvg.

Table 4. Mean determinant sharpness (DS) of the predictive distributions calculated with $\text{FedAG}_{N=1}^{(t=5)}$ and DE.

Dataset	$\textbf{FedAG}_{N=1}^{(t=5)}$	DE
Boston	4.05	4.79
Concrete	6.37	7.07
Energy	2.16	2.67
Kin8nm	0.10	0.14
Naval Propulsion	0.01	0.02
Power Plant	4.30	5.48
Protein	3.88	4.59
Red Wine	0.79	0.69
Yacht	2.85	0.94
Year Prediction	11.12	7.85

4.6. Discussion

As each of the clients in FedAG only has access to a fraction of the dataset, we do not expect it to out-perform the benchmarks BLR, VI and DE, which simultaneously have access to the complete dataset. Nevertheless, the linear models BLR and $\text{FedAG}_{N=0}$ attain almost identical performance. Consequently, $\text{FedAG}_{N=0}$ can be applied as an alternative to BLR in federated, distributed settings. In the case of a non-linear model, the performance of $\text{FedAG}_{N=1}$ can generally compete with that of VI and approaches the performance of DE on some datasets. Additionally, the sharpness of the predictive distributions calculated with FedAG and DE is comparable. Hence, $\text{FedAG}_{N=1}$ can be used as a probabilistic model in a federated setting, achieving predictive performance comparable with state-of-the-art non-federated and non-distributed methods. Further advantages of FedAG are the retained privacy and communication efficiency [48]. Therefore, FedAG offers prediction performance comparable with state-of-the art probabilistic ML algorithms in an efficient and privacy-preserving manner.

5. Conclusions and Future Work

Interest in the application of ML algorithms in distributed or federated settings is increasing. There, predictive uncertainty is an important feature that needs to be addressed. We presented FedAvg-Gaussian (FedAG), an efficient method for the learning of probabilistic models in a distributed, federated setting. FedAG extends FedAvg to include predictive uncertainty, by treating the set of local weights as a posterior distribution for the weights of the global model. Therefore, predictive uncertainty can be represented in a computation- and communication-efficient way, so that probabilistic on-device machine learning is realized.

We used FedAG to learn two different models, a linear regression and a feed-forward neural network with a single hidden layer. The performance of our method was evaluated on UCI regression datasets and compared to benchmark methods using proper scoring rules. When implemented with a linear regression model, FedAG's performance is similar to that of a BLR. FedAG with a neural network can after $t = 5$ communication rounds outperform VI on most datasets and its performance approaches that of DE on several datasets.

Our future work includes several topics. FedAG could benefit from an aggregation step more robust to outliers and adversarial attacks, such as in the methods Krum [49] and Aggregathor [50]. Moreover, further work should aim to test the methods with real federated data to analyze the effect of non-IID partitioning and stratified splits compared to randomized splits [51]. For the personalization of the local models, different aggregation and initialization methods can be applied. Evaluating such concepts in an asynchronous federated learning environment might prove an important area for future research. Furthermore, larger neural networks and other deep learning architectures such as convolutional neural networks can be applied and analyzed. Finally, our future research will aim to benchmark FedAG against other novel federated learning concepts, such as partitioned variational inference (PVI) [34], distributed Stein variational gradient descent (DSVGD) [36], federated posterior averaging (FedPA) [37], and the combination of FedAvg and Monte Carlo dropout [38].

Author Contributions: Methodology, A.T.T.; Software, A.T.T.; Validation, A.T.T.; Writing—original draft preparation, A.T.T.; writing—review and editing, A.T.T.; supervision, F.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

Acknowledgments: We acknowledge support by the KIT-Publication Fund of the Karlsruhe Institute of Technology.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Véstias, M.P.; Duarte, R.P.; de Sousa, J.T.; Neto, H.C. Moving Deep Learning to the Edge. *Algorithms* **2020**, *13*, 125. [CrossRef]
2. Zinkevich, M.A.; Weimer, M.; Smola, A.; Li, L. Parallelized Stochastic Gradient Descent. In Proceedings of the 23rd International Conference on Neural Information Processing Systems (NIPS'10), Vancouver, BC, Canada, 6–9 December 2010; Curran Associates, Inc.: Red Hook, NY, USA, 2010; Volume 2, pp. 2595–2603.
3. Jaggi, M.; Smith, V.; Takac, M.; Terhorst, J.; Krishnan, S.; Hofmann, T.; Jordan, M.I. Communication-Efficient Distributed Dual Coordinate Ascent. In *Advances in Neural Information Processing Systems 27*; Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2014; pp. 3068–3076.
4. Ormándi, R.; Hegedüs, I.; Jelasity, M. Efficient P2P Ensemble Learning with Linear Models on Fully Distributed Data. *arXiv* **2011**, arXiv:1109.1396.
5. McMahan, H.B.; Moore, E.; Ramage, D.; y Arcas, B.A. Federated Learning of Deep Networks using Model Averaging. *arXiv* **2016**, arXiv:1602.05629.

6. Neal, R.M. *Bayesian Learning for Neural Networks*; Springer Science & Business Media: New York, NY, USA, 2012; Volume 118.
7. Lakshminarayanan, B.; Pritzel, A.; Blundell, C. Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17), Long Beach, CA, USA, 4–9 December 2017; Curran Associates, Inc.: Red Hook, NY, USA, 2017; pp. 6405–6416.
8. Gneiting, T.; Katzfuss, M. Probabilistic Forecasting. *Annu. Rev. Stat. Its Appl.* **2014**, *1*, 125–151. [CrossRef]
9. Lee, E.; Hall, J.; Meadowcroft, I. Coastal cliff recession: The use of probabilistic prediction methods. *Geomorphology* **2001**, *40*, 253–269. [CrossRef]
10. Brusaferri, A.; Matteucci, M.; Portolani, P.; Vitali, A. Bayesian deep learning based method for probabilistic forecast of day-ahead electricity prices. *Appl. Energy* **2019**, *250*, 1158–1175. [CrossRef]
11. Cutore, P.; Campisano, A.; Kapelan, Z.; Modica, C.; Savic, D. Probabilistic prediction of urban water consumption using the SCEM-UA algorithm. *Urban Water J.* **2008**, *5*, 125–132. [CrossRef]
12. Xie, W.; Zhang, P.; Chen, R.; Zhou, Z. A Nonparametric Bayesian Framework for Short-Term Wind Power Probabilistic Forecast. *IEEE Trans. Power Syst.* **2019**, *34*, 371–379. [CrossRef]
13. Hu, Y.; Zhan, W.; Tomizuka, M. Probabilistic Prediction of Vehicle Semantic Intention and Motion. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018; pp. 307–313. [CrossRef]
14. Fünfgeld, S.; Holzäpfel, M.; Frey, M.; Gauterin, F. Stochastic Forecasting of Vehicle Dynamics Using Sequential Monte Carlo Simulation. *IEEE Trans. Intell. Veh.* **2017**, *2*, 111–122. [CrossRef]
15. Scheubner, S.; Thorgeirsson, A.T.; Vaillant, M.; Gauterin, F. A Stochastic Range Estimation Algorithm for Electric Vehicles Using Traffic Phase Classification. *IEEE Trans. Veh. Technol.* **2019**, *68*, 6414–6428. [CrossRef]
16. Ashukha, A.; Lyzhov, A.; Molchanov, D.; Vetrov, D. Pitfalls of In-Domain Uncertainty Estimation and Ensembling in Deep Learning. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020.
17. Bishop, C. *Pattern Recognition and Machine Learning*; Springer: New York, NY, USA, 2006.
18. Dera, D.; Rasool, G.; Bouaynaya, N. Extended Variational Inference for Propagating Uncertainty in Convolutional Neural Networks. In Proceedings of the 2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP), Pittsburgh, PA, USA, 13–16 October 2019; pp. 1–6. [CrossRef]
19. Wang, P.; Bouaynaya, N.C.; Mihaylova, L.; Wang, J.; Zhang, Q.; He, R. Bayesian Neural Networks Uncertainty Quantification with Cubature Rules. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–7.
20. Wang, H.; Yeung, D.Y. A Survey on Bayesian Deep Learning. *ACM Comput. Surv.* **2020**, *53*, 1–37. [CrossRef]
21. Graves, A. Practical Variational Inference for Neural Networks. In Proceedings of the 24th International Conference on Neural Information Processing Systems (NIPS'11), Atlanta, GA, USA, 11–15 July 2011; Curran Associates, Inc.: Red Hook, NY, USA, 2011; pp. 2348–2356.
22. Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; Wierstra, D. Weight Uncertainty in Neural Network. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; Bach, F., Blei, D., Eds.; PMLR: Lille, France, 2015; Volume 37; pp. 1613–1622.
23. Hernández-Lobato, J.M.; Adams, R.P. Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. In Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML'15), Lille, France, 6–11 July 2015; PMLR: Lille, France, 2015; Volume 37, pp. 1861–1869.
24. Gal, Y.; Ghahramani, Z. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 20–22 June 2016; Balcan, M.F., Weinberger, K.Q., Eds.; PMLR: New York, NY, USA, 2016; Volume 48. pp. 1050–1059.
25. Maddox, W.J.; Izmailov, P.; Garipov, T.; Vetrov, D.P.; Wilson, A.G. A Simple Baseline for Bayesian Uncertainty in Deep Learning. In *Advances in Neural Information Processing Systems*; Wallach, H., Larochelle, H., Beygelzimer, A., d'Alch'e-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Vancouver, BC, Canada, 2019; Volume 32. pp. 13153–13164.
26. Bröcker, J.; Smith, L.A. From ensemble forecasts to predictive distribution functions. *Tellus A* **2008**, *60*, 663–678. [CrossRef]
27. Wilks, D.S. Smoothing forecast ensembles with fitted probability distributions. *Q. J. R. Meteorol. Soc.* **2002**, *128*, 2821–2836. [CrossRef]
28. Baran, S.; Lerch, S. Combining predictive distributions for the statistical post-processing of ensemble forecasts. *Int. J. Forecast.* **2018**, *34*, 477–496. [CrossRef]
29. Leutbecher, M.; Palmer, T. Ensemble forecasting. *J. Comput. Phys.* **2008**, *227*, 3515–3539. [CrossRef]
30. Huang, G.; Li, Y.; Pleiss, G.; Liu, Z.; Hopcroft, J.E.; Weinberger, K.Q. Snapshot Ensembles: Train 1, get M for free. *arXiv* **2017**, arXiv:1704.00109.
31. Garipov, T.; Izmailov, P.; Podoprikhin, D.; Vetrov, D.; Wilson, A.G. Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs. In Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18), Montreal, QC, Canada, 3–8 December 2018; Curran Associates, Inc.: Red Hook, NY, USA, 2018; pp. 8803–8812.
32. Antorán, J.; Urquhart Allingham, J.; Hernández-Lobato, J.M. Depth Uncertainty in Neural Networks. *arXiv* **2020**, arXiv:2006.08437.
33. Ustimenko, A.; Prokhorenkova, L.; Malinin, A. Uncertainty in Gradient Boosting via Ensembles. *arXiv* **2020**, arXiv:2006.10562.
34. Bui, T.D.; Nguyen, C.V.; Swaroop, S.; Turner, R.E. Partitioned Variational Inference: A unified framework encompassing federated and continual learning. *arXiv* **2018**, arXiv:1811.11206.

35. Sharma, M.; Hutchinson, M.; Swaroop, S.; Honkela, A.; Turner, R.E. Differentially Private Federated Variational Inference. *arXiv* **2019**, arXiv:1911.10563.
36. Kassab, R.; Simeone, O. Federated Generalized Bayesian Learning via Distributed Stein Variational Gradient Descent. *arXiv* **2020**, arXiv:2009.06419.
37. Al-Shedivat, M.; Gillenwater, J.; Xing, E.; Rostamizadeh, A. Federated Learning via Posterior Averaging: A New Perspective and Practical Algorithms. *arXiv* **2020**, arXiv:2010.05273.
38. Zhang, X.; Fang, F.; Wang, J. Probabilistic Solar Irradiation Forecasting based on Variational Bayesian Inference with Secure Federated Learning. *IEEE Trans. Ind. Inform.* **2020**. [CrossRef]
39. Xiao, P.; Cheng, S.; Stankovic, V.; Vukobratovic, D. Averaging Is Probably Not the Optimum Way of Aggregating Parameters in Federated Learning. *Entropy* **2020**, *22*, 314. [CrossRef]
40. Wilson, A.G.; Izmailov, P. Bayesian Deep Learning and a Probabilistic Perspective of Generalization. *arXiv* **2020**, arXiv:2002.08791.
41. Geyer, C. Introduction to Markov Chain Monte Carlo. In *Handbook of Markov Chain Monte Carlo*; CRC Press: Boca Raton, FL, USA, 2011; p. 45.
42. Mooney, C.F.; Mooney, C.L.; Mooney, C.Z.; Duval, R.D.; Duvall, R. *Bootstrapping: A Nonparametric Approach to Statistical Inference*; Number 95; Sage: London, UK, 1993.
43. Jordan, A.; Krüger, F.; Lerch, S. Evaluating Probabilistic Forecasts with scoringRules. *J. Stat. Softw.* **2019**, *90*. [CrossRef]
44. Gneiting, T.; Raftery, A.E. Strictly Proper Scoring Rules, Prediction, and Estimation. *J. Am. Stat. Assoc.* **2007**, *102*, 359–378. [CrossRef]
45. Dua, D.; Graff, C. UCI Machine Learning Repository. 2017. Available online: https://archive.ics.uci.edu/ml/index.php (accessed on 19 July 2020).
46. Mukhoti, J.; Stenetorp, P.; Gal, Y. On the Importance of Strong Baselines in Bayesian Deep Learning. *arXiv* **2018**, arXiv:1811.09385.
47. Choromanska, A.; Henaff, M.; Mathieu, M.; Arous, G.B.; LeCun, Y. The Loss Surfaces of Multilayer Networks. In Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, San Diego, CA, USA, 9–12 May 2015; Lebanon, G., Vishwanathan, S.V.N., Eds.; PMLR: San Diego, CA, USA, 2015; Volume 38, pp. 192–204.
48. Asad, M.; Moustafa, A.; Ito, T. FedOpt: Towards Communication Efficiency and Privacy Preservation in Federated Learning. *Appl. Sci.* **2020**, *10*, 2864. [CrossRef]
49. Blanchard, P.; El Mhamdi, E.M.; Guerraoui, R.; Stainer, J. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *Advances in Neural Information Processing Systems 30*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2017; pp. 119–129.
50. Damaskinos, G.; El-Mhamdi, E.M.; Guerraoui, R.; Guirguis, A.; Rouault, S. AGGREGATHOR: Byzantine Machine Learning via Robust Gradient Aggregation. In Proceedings of the Machine Learning and Systems 2019, Stanford, CA, USA, 31 March–2 April 2019; pp. 81–106.
51. Caldas, S.; Wu, P.; Li, T.; Konecný, J.; McMahan, H.B.; Smith, V.; Talwalkar, A. LEAF: A Benchmark for Federated Settings. *arXiv* **2018**, arXiv:1812.01097.