CMSC 828A Final Report

Daeun Jung, Kyungyeon Lee, Evar Jones, Cemil Nureddin Vahapoglu, and **Yoon Kyung Shon**
{daeunj, kylee, ejones26, cemilnv, ykshon}@umd.edu

1 Introduction

Federated learning is a powerful technique used in machine learning, which allows multiple clients to be trained simultaneously on their own unique dataset. It attempts to reduce communication costs by alternating local training with global aggregation of model parameters without ever directly sharing client data.

Synchronous federated learning is an algorithm where the global model is aggregated from each client's model for every round. The underlying assumption of this algorithm is that each client sends their model once local training is complete and then receives the updated global model. As a result, the client may lose their trained local model and obtain the global model, which may exhibit lower performance than their model.

In `FedAvg` (McMahan et al., 2023), a global model combines local stochastic gradient descent (SGD) on each client with model averaging to produce a more generalized global model. `FedProx` (Li et al., 2020) can handle a heterogeneous federated environment by including a penalization term to prevent the local model from deviating too far from the global model.

Rather than focusing solely on learning a global model, we utilize a federated learning approach which allows for the development of personalized models for each client, while also creating a generalized global model. This is achieved by providing the opportunity to select the degree of updating. Our contributions are as follows:

- We conduct extensive experiments and analysis of our novel algorithmic framework using the CIFAR-10 and CIFAR-100 datasets, exploring different hyperparameter search techniques within a federated learning setting.

- Our experiments reveal that individually tuned FedAvg and FedProx approaches exhibit limitations, and performs worse than our algorithmic approach that essentially combines their methods, while optimizing their hyperparameters.

2 Problem Formulation

In our new federated learning approach, we improve client selection and reweighing. We try to find the hyperparameters that control the client and global models formulated as follows:

$$\begin{aligned} \min_{\alpha,\beta} \quad & \textstyle\sum_{j=1}^{m} F_j^{val}(\theta_j) \\ s.t. \quad & \theta_i \in_{\theta\in\mathbb{R}^d} F_i^{train}(\theta) + \tfrac{\beta_i}{2}\|\theta - \theta_0\|_2^2 \\ & \theta_0 \in_{\theta\in\mathbb{R}^d} \textstyle\sum_{j=1}^{m} \alpha_j F_j^{train}(\theta), \\ & \text{where}, \forall\ i \in [m]. \end{aligned} \qquad (1)$$

In federated learning, we typically have a global model θ_0, and local models $\theta_{1:m}$ owned by each client. The objective function is defined as the sum of validation loss from each client model θ_j. To regulate the aggregation of the client model parameter, we assign two hyperparameter vectors, α and β, to clients and global models.

The α vector serves as a weight vector of the global model, which mediates when aggregating the client model parameter. By maximizing efficiency, the global model with the α vector regulates the trade-off between all local losses, while $\beta_{1:m}$ controls the trade-off between the local training and global alignment. For instance, in `FedAvg`, α can be set to $\frac{1}{m}$ or $\frac{1}{p_i}$, where each client has p_i number of data, for unbalanced data.

Clients are given the freedom to choose how much they will update between their trained local model and updated global model by deciding β. Unlike `FedAvg`, which prevents local model trains from the global model, β manages both reparameterization and personalization. Therefore, $\theta_{0:m}$ can decide how much each model should be updated based on α or β. This approach provides greater flexibility and control for clients, enabling them to personalize their models while preserving the global model's accuracy.

Our method is related to meta-learning algorithms used in training neural networks that can generalize to novel tasks. In federated learning, we aim to develop both a personalized model and a generalized global model simultaneously. Thus, our objective function is to minimize the validation loss of the clients. To achieve this goal, we carefully examine the hyperparameters for both clients and servers. This is crucial because the initial hyperparameters significantly impact the performance gains in federated learning compared to local SGD training.

To set these hyperparameters, we utilized random search and Bayesian optimization techniques. These methods helped us to efficiently and effectively find the optimal hyperparameters needed to achieve the desired results.

3 Research Methods

Several methods have been developed to tackle hyperparameter searching, including a basic search technique such as random search, as well as a more automated search technique such as Bayesian optimization.

Random search is a hyper-parameter optimization that performs random sampling of hyperparameters from the search space. Since random search does not depend the gradients of the objective function of the problem, it can be used for non-continuous and non-differentiable functions as well. However, the performance of the random search highly depends on the structure of the search space. Achieving a good configuration can be hard. Therefore, different variants of random search are proposed in the literature to have some structured ways of searching although it is random. Those variants are mostly depend on how to decide the step size of the searching. Fixed Step Size Random Search (FSSRS), Adaptive Step Size Random Search (ASSRS), Optimized Relative Step Size Random Search (ORSSRS) can be given as some examples for those variants.

It has been showed that random search is more time-efficient than grid search hyper-parameter optimization (Bergstra and Bengio, 2012). However, it can also be computationally expensive when the search space is large. In federated learning setting, it can be run on each client to determine β_i values in equation 1 for each client. Additionally, it can run on the central server for α values to achieve the best global model θ_0. Random search is more efficient than grid search. However, neither of these techniques learn anything from previous experiments as the model's training on one set of hyperparameters is performed in isolation with the other set of hyperparamters. Additionally, a high number of hyperparameters could result in these techniques becoming intractable or not feasible. These techniques also require a lot of time and computational resources. As a result, there is a lot risk in these techniques, with the additional uncertainty that they will give the best set of hyperparameters.

Bayesian optimization (BO) is a hyperparameter search method that attempts to solve these challenges. It is a hyperparameter search technique that uses the concept of Bayes theorem to guide the search to minimize or maximize an objective function. In BO, a surrogate model is built for the objective and quantifies the uncertainty in that surrogate using Gaussian process regression, and then uses an acquisition function defined from this surrogate to decide where to sample (Frazier, 2018). After each experiment, the surrogate model learns and becomes more confident in choosing the next set of hyperparameters that should be experimented with, resulting in a better performance.

Hyperparameter optimization approaches, such as Bayesian optimization, have been investigated before in federated learning systems. Federated Bayesian optimization (FBO) was introduced to extend Bayesian optimization to the federated learning setting (Dai et al., 2020). Another study investigated a local hyperparameter optimization approach using Bayesian optimization, that in contrast to a global hyperparameter optimization approach, allows each client to have its own hyperparameter configuration (Holly et al., 2021). Although general hyperparameter optimization has been the subject of intense study, tuning hyperparameters can be especially challenging. Our work attempts to extend these search methods as an approach to find optimal hyperparameters for better client selection and reweighing.

Stochastic gradient descent (SGD) can also be evaluated as the method to find α and β values in equation 1 since gradient descent is an optimization technique preferred highly for machine learning settings. However, it should be noted SGD is used to optimize non-constrained optimization problems. Therefore, feasible set of the non-constrained problem should be obtained first with respect to the constraints of the problem in equation 1. Then, SGD should be applied on it.

In this work, we investigate the impact of these hyperparameter optimization approaches using the Cifar datasets, with a ResNet18 baseline.

4 Related Work

Several improvements have been proposed to `FedAvg` (McMahan et al., 2023) address the challenges of system heterogeneity and statistical heterogeneity in synchronous federated learning. However, this approach assumes that the data on each device is the identically and independently distributed(IID), which may not hold in practice. To address this, `FedDane` (Li et al., 2019) is proposed, which uses a data-dependent regularization term to account for non-IID data in federated learning. `FedProx` (Li et al., 2020) extends `FedAvg` by incorporating a proximal term that encourages devices to remain close to their local model parameters. This method has been shown to improve convergence speed and performance on non-IID data. Another approach is to adjust the learning rate based on the number of local iterations performed by each device. `FedAdapt` (Wu et al., 2022) dynamically adjusts the learning rate based on the local iteration count, and has shown to outperform `FedAvg` and `FedProx` in a range of federated learning scenarios.

Traditional methods of federated learning have shortcomings when it comes to fairness and robustness. Robustness relates to the model's ability to maintain its performance even when there are device failures or malicious actors attempting to manipulate the training process.

The `Ditto` (Li et al., 2021), a multi-task learning objective for federated learning that provides personalization leverages each device's individual data distribution to improve fairness and robustness. This approach customizes the model for each device based on its unique data distribution and then aggregates the personalized models to form a global model while retaining similar efficiency and privacy benefits as traditional federated learning. Finally, other works have explored the use of personalized models in federated learning. For instance, `sketched updates` (Smith et al., 2017) proposed personalized federated learning, which trains a separate model for each device using its local data. However, this approach requires more communication and computation resources than traditional federated learning.

Drawing upon previous research, we have developed a method to improve both personalization and generalization by adapting the values of α and β. Our approach bears similarity to `Ditto`, but differs in that we utilize a server solver to optimize the global model using the `FedAvg` method. Instead of assigning identical weights to each client, our algorithm learns a global model that aggregates locally trained models with different weights, based on the congruence of the global model. By updating two parameter vectors, our FL algorithm can learn both personalized models using individual data and a generalized global model.

5 Experiments & Results

5.1 Expected Results

The expected result of this study is to demonstrate the improved performance of our proposed federated learning model, which combines the FedAvg and FedProx algorithms. We compare the performance of our model to both the FedProx and FedAvg baselines, as well as a local SGD model, in order to evaluate its effectiveness.

We anticipate that our model will outperform the baselines and show superior performance compared to the local SGD model. This expectation is based on the fact that our model incorporates the strengths of both FedAvg and FedProx, leveraging the advantages of each algorithm to enhance the overall performance of federated learning.

Furthermore, we expect that our model's performance will be influenced by the values assigned to the hyperparameters alpha and beta. We desire larger alpha values for models with lower validation loss, indicating a higher weight assigned to models that perform better. In contrast, for models with a smaller amount of data, we prefer larger beta values. This preference indicates that clients with fewer data prioritize the global model during training rather than relying on personalized models.

5.2 Experiments Setting

Regarding the experiment setting, we conducted synchronous Federated Learning using CIFAR-10 and CIFAR-100 datasets, which were allocated at 5 levels among the clients. This allocation scheme aimed to provide a diverse and representative distribution of data across the participating clients.

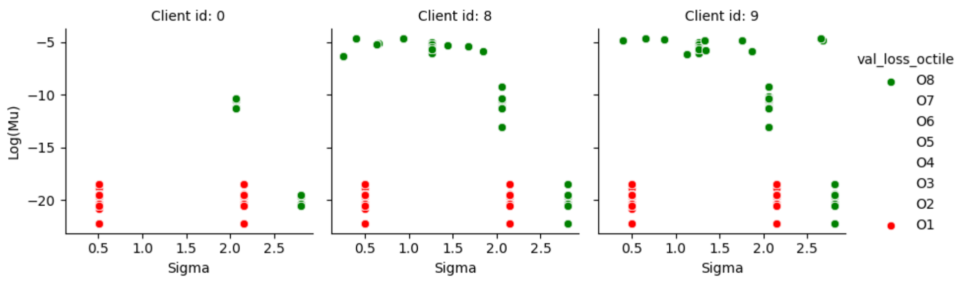


Figure 1: Random search result using Cifar10 dataset

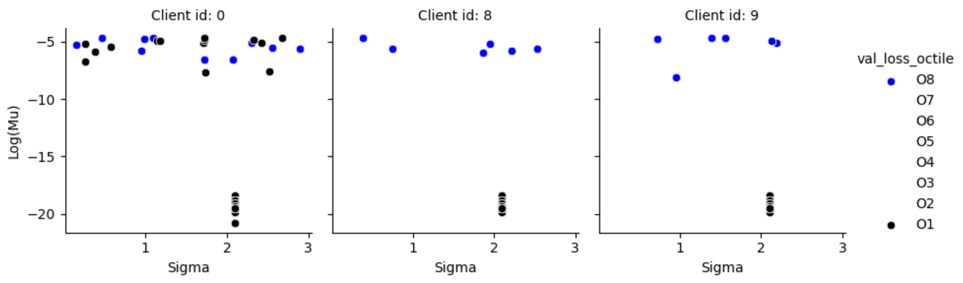


Figure 2: Random search result using Cifar100 dataset

For Baseline 1, we fixed the global round to 1 and conducted 200 local epochs, while in Baseline 2, we fine-tuned the learning rate, weight decay, and momentum values to establish an optimized performance baseline.

For Baseline 2, the number of global round is 20 and the number of local round is 20 during fine-tuning process of learning rate, weight decay, and momentum coefficients. For tuning of those hyperparameters, Optuna framework is utilized. The maximization of the evaluation accuracy is considered for it.

By Optuna framework, 4 different hyperparameter tuning processes are performed: FedAvg for Cifar10, FedAvg for Cifar100, FedProx for Cifar10, and FedProx for Cifar100. In each process, 100 trials are conducted to obtain a good configuration of learning rate, weight decay and momentum. Additionally, TPE Sampler and Median Pruner are utilized for trials to be able to have time efficiency during trials.

All code, data and experiments are publicly available at: `https://github.com/daeun-j/CMSC828A.git`

5.3 Random Search Experiments

To implement the random search method, we first randomly selected sigma and mu values based on the Gaussian distribution. These values were then assigned to both the client and the server, generating random alpha and beta values. By applying the random search method, we obtained two sets of optimal hyperparameter vectors, alpha and beta, for the CIFAR-10 and CIFAR-100 datasets. For the CIFAR-10 dataset, the optimal value of μ was randomly selected from between 1e-2 and 1e-4. The σ value was chosen randomly from between 0.1 and 3. On the other hand, for the CIFAR-100 dataset, μ was randomly selected from between 1e-1 and 1e-2, while σ was chosen from the same range as before, between 0.1 and 3.

The experiments involved 10 clients, each characterized by different data characteristics. For example, Client 0 had a small amount of data, while Client 9 had a large amount. We partitioned the validation loss into eight distinct octiles to examine extreme scenarios where the validation loss is at its lowest and highest points. In Figure1 and 2, the color-coded graphs represented the loss rates, where Q8 indicated the lowest validation loss (good performance) and Q1 indicated the highest validation loss (poor performance).

In Figure1, focusing on Client 9 with the CIFAR-10 dataset, we observed a higher concentration of green dots (representing smaller validation loss) towards the bottom compared to Client 0. This suggests that Client 9 prefers a smaller mu, leading to a smaller beta value, resulting in lower validation loss. This preference aligns with the fact that Client 9 has a large amount of data, indicating less reliance on high beta values.

In Figure2, examining Client 0 with the CIFAR-100 dataset, we noticed a broader distribution of blue dots (indicating lower validation loss) at larger mu values. This indicates that due to larger mu, the beta value increases, implying that the local model of Client 0 is influenced by the global model. This observation aligns with the fact that Client 0 has a small amount of data, making it more reliant on the global model for training.

Based on the results obtained from the six graphs, we consistently observed that Q1 (represented by black and red dots indicating high validation loss) consistently appeared at the bottom (corresponding to smaller mu values). This implies that all clients exhibited a similar tendency of preferring smaller mu values and smaller beta values. This finding demonstrates that in the problem formulation, smaller beta values correspond to personalized training. Moreover, when clients prioritize personalization, there is a possibility of higher validation loss.

5.4 Bayesian Optimization

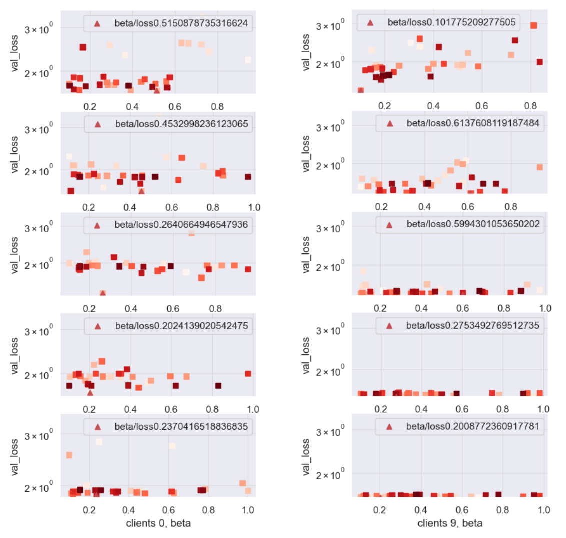
To optimize the hyperparameters in our proposed model, we utilized the hyperparameter values found in Baseline 2 as a starting point. By leveraging Bayesian optimization, we aimed to further enhance the performance of our model by fine-tuning the values of alpha and beta.

First, we focused on finding the optimal value for beta while keeping alpha fixed, following the FedProx approach. Subsequently, we fixed the beta value and searched for the optimal alpha value using the FedAvg approach.

Figure 3a illustrates the search for the optimal beta values over time. The horizontal axis represents different beta values, while the vertical axis represents the corresponding validation loss. The graph shows the progression of the model's performance as it trains. Beta value for Client 0 is chosen among 0.5-0.7 for its initial stability. Bayesian optimization is adaopted to online-learning scheme, the models are trained as time goes by. To minimize the effectiveness of overtime, we select the best beta from the early stage of beta values.

Dataset	α	β
Cifar10	0.04	0.1-0.4(# data $\uparrow$) / 0.5-07(# data $\downarrow$)
Cifar100	0.05	0.01-0.02(# data $\uparrow$) / 0.06-0.08(# data $\downarrow$)

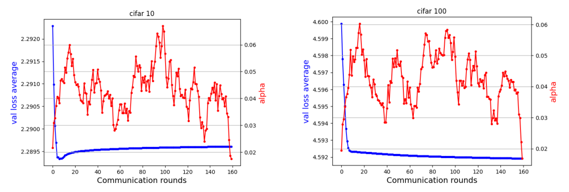
Table 1: Optimized hyper-parameters (α, β) for each dataset



(a) Client 0 (400 training data)

(b) Client 9 (1200 training data)

Figure 3: Bayesian optimization on β



(a) Cifar 10

(b) Cifar 100

Figure 4: Bayesian optimization on α

Observations from the graph revealed that a higher beta value for Client 0, which had a smaller dataset, led to a lower validation loss, indicating that prioritizing the global model over personalized models yielded better results. Conversely, for clients with larger datasets, in Figure 3b, a lower beta value resulted in improved performance. This observation aligns with the FedProx approach, where the weighting of personalized models is adjusted based on the amount of data available.

Figure 4 shows the alpha values used for CIFAR-10 and CIFAR-100. The alpha value was determined at the point where the validation loss experienced a sudden drop. This substantial decrease indicated a significant improvement in the model's performance and was considered the optimal alpha value for the respective dataset. The final optimal α and β values are summarised on Table 1.

5.5 Stochastic Gradient Methods

SGD updates the parameters using only a single training instance in each iteration. This method provides fast convergence with noisy estimates of

Methods	Test Acc(Cifar10)	Test Acc(Cifar100)	Training Time (Cifar10)	Training Time (Cifar100)	Big ($\mathcal{O}$)
(Baseline 1)	31.826	27.602	5'47"	5'46"	$\mathcal{O}(kn^2)$
FedAvg (Baseline 2)	74.69	43.02	41'08"	44'05"	$\mathcal{O}(Mkn^2)$
FedProx (Baseline 2)	74.60	46.36	45'44"	46'15'	$\mathcal{O}(Mkn^2)$
SGD	75.71	52.72	142'32"	144'43"	$\mathcal{O}(kn^2)$
Random Search	73.69	52.30	360'52"	180'27"	$\mathcal{O}(nlogn)$
Bayesian Optimization	75.11	53.02	121'43"	144'45"	$\mathcal{O}(n^2d^2)$

Table 2: Performance comparison of our algorithm using hyper-parameter search methods

the error gradient. The result obtained by differentiating the objective function shows that β and α are continuously calculated over time. We omitted the differentiation equation of the objective function, including validation loss. SGD method searches hyperparameters alternatively.

5.6 Results and Analysis

To evaluate the performance of our proposed model, we compared it against the FedAvg and FedProx methods, as well as a baseline local SGD as shown in Figure 5. Results showed that our model demonstrated significant improvements in terms of performance for CIFAR-100 dataset. However, it is important to note that Bayesian optimization and SGD methods achieved higher test accuracy for both CIFAR-10 and CIFAR-100 datasets. Federated learning yields superior performance compared to local SGD, highlighting the advantages of client involvement in FL. This observation suggests that while our proposed model showed promise in enhancing the performance of federated learning, further exploration and optimization may be required to achieve better results for CIFAR-10.

Time complexity for each algorithm is shown in Table 2. In the case of SGD, it shows a fairly high complexity, because time complexity highly depend on updating hyperparameters and training with the meta-learning method such as validation evaluation. Bayesian optimization can be controlled by determining the number of searching points. n is the number of samples on which the surrogate function is evaluated and d is the number of hyperparameters. We set n = 3, d = 20. If we increase n higher to ensure that the variance is small, the time complexity will increase quadratic.

To compare the time complexity with local SGD, where k is the number of features and n is the total number of data points, the FedAvg, and FedProx follows the local SGD with multiplying the global rounds M.

6 Limitations

When we utilized random search with a Gaussian distribution, we observed a significant disparity in accuracy between the model's performance on the CIFAR-10 and CIFAR-100 datasets. This finding suggests that our model requires further generalization, achieved through the use of more diverse datasets. Moreover, it was challenging to identify any clear preference for specific beta values among clients based on the validation loss alone. As a result, we decided to divide the validation loss into eight partitions instead of four. By doing so, we aim to enhance the visibility of distinct beta values that are favored by different clients. To achieve this, we plan to amplify the differences in the datasets utilized by each client, allowing for more pronounced variations in the validation loss.

Given that we utilized 10 clients, we had to obtain 20 hyperparameters. It would have been preferable to employ multi-parameter search methods for this task. Moreover, it is believed that better results could have been achieved if the search had been conducted with a larger number of samples. Even after training conducted, the hyperparameters still shows considerably high variance. Additionally, while we opted for the online search method, it is crucial to consider whether this approach is suit-

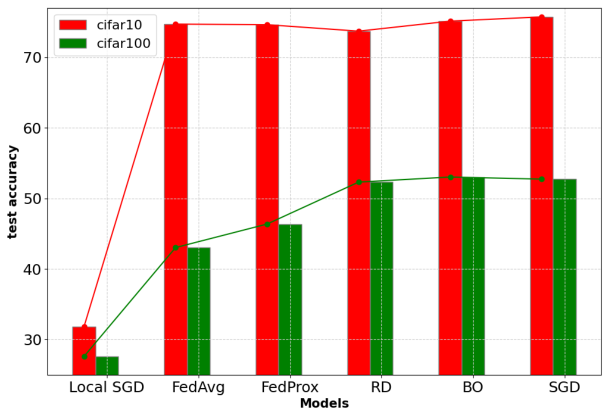


Figure 5: Test accuracy comparison to our methods on Cifar10 and Cifar100

able for federated learning. Bayesian optimization is appropriate for determining the β value when focusing solely on the client's perspective. However, it is considered which scheme is better between repeatedly learning both the α and β values and to alternatively learning after searching for the α value with the optimized β and then fixing it.

We examine our method under limited FL setting, such as IID-setting with full-participation. As each client possesses a distinct data distribution, the requirement for selecting a more granular β becomes more pronounced. To create a more realistic FL setting, adjusting the number of participants, we anticipate that the server will become more sensitive to the choice of model using α. Furthermore, future research directions will encompass exploring the security and privacy aspects, which are inherent advantages of FL, as well as investigating mathematical convergence proofs.

7 Conclusion

In conclusion, our study has demonstrated the superior performance of our novel algorithm. This algorithm, which ingeniously combines the FedAvg and FedProx methods while simultaneously optimizing hyperparameters, has outpaced the performance of the individually tuned FedAvg and FedProx approaches within the context of federated learning. The comparative performance advantages of our algorithm over traditional methods are clearly illustrated in Figure 5. The results aligned with our expectations and provided strong evidence for the effectiveness of our approach. These findings highlight the potential of our algorithm to address challenges in federated learning and improve overall performance, although further exploration is needed to enhance its performance across different datasets.

References

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305.

Zhongxiang Dai, Bryan Kian Hsiang Low, and Patrick Jaillet. 2020. Federated bayesian optimization via thompson sampling.

Peter Frazier. 2018. A tutorial on bayesian optimization.

Stephanie Holly, Thomas Hiessl, Safoura Rezapour Lakani, Daniel Schall, Clemens Heitzinger, and Jana Kemnitz. 2021. Evaluation of hyperparameter-optimization approaches in an industrial federated learning system.

Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. 2021. Ditto: Fair and robust federated learning through personalization.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smithy. 2019. Feddane: A federated newton-type method. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 1227–1231. IEEE.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2023. Communication-efficient learning of deep networks from decentralized data.

Virginia Smith, Eric Chiang, Mehrdad Sanjabi, and Martin Jaggi. 2017. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*.

Di Wu, Rehmat Ullah, Paul Harvey, Peter Kilpatrick, Ivor Spence, and Blesson Varghese. 2022. Fedadapt: Adaptive offloading for iot devices in federated learning. *IEEE Internet of Things Journal*, 9(21):20889–20901.