WAFFLe: Weight Anonymized Factorization for Federated Learning

Weituo Hao[1], **Nikhil Mehta**[1], **Kevin J Liang**[1], **Pengyu Cheng**[1],
Mostafa El-Khamy[2], **Lawrence Carin**[1]
[1]Duke University, [2]Samsung
weituo.hao@duke.edu

Abstract

In domains where data are sensitive or private, there is great value in methods that can learn in a distributed manner without the data ever leaving the local devices. In light of this need, federated learning has emerged as a popular training paradigm. However, many federated learning approaches trade transmitting data for communicating updated weight parameters for each local device. Therefore, a successful breach that would have otherwise directly compromised the data instead grants whitebox access to the local model, which opens the door to a number of attacks, including exposing the very data federated learning seeks to protect. Additionally, in distributed scenarios, individual client devices commonly exhibit high statistical heterogeneity. Many common federated approaches learn a single global model; while this may do well on average, performance degrades when the i.i.d. assumption is violated, underfitting individuals further from the mean, and raising questions of fairness. To address these issues, we propose Weight Anonymized Factorization for Federated Learning (WAFFLe), an approach that combines the Indian Buffet Process with a shared dictionary of weight factors for neural networks. Experiments on MNIST, FashionMNIST, and CIFAR-10 demonstrate WAFFLe's significant improvement to local test performance and fairness while simultaneously providing an extra layer of security.

1 Introduction

With the rise of the Internet of Things (IoT), the proliferation of smart phones, and the digitization of records, modern systems generate increasingly large quantities of data. These data provide rich information about each individual, opening the door to highly personalized intelligent applications, but this knowledge can also be sensitive: images of faces, typing histories, medical records, and survey responses are all examples of data that should be kept private. Federated learning [21] has been proposed as a possible solution to this problem. By keeping user data on each local *client* device and only sharing model updates with the global *server*, federated learning represents a possible strategy for training machine learning models on heterogeneous, distributed networks in a privacy-preserving manner. While demonstrating promise in such a paradigm, a number of challenges for federated learning [16] remain.

As with centralized distributed learning settings [4], many federated learning algorithms focus on learning a single global model. However, due to variation in user characteristics or tendencies, personal data are highly likely to exhibit significant *statistical heterogeneity*. To simulate this, federated learning algorithms are commonly tested in non-i.i.d. settings [21, 29, 36, 15, 26], but data are often equally represented across clients and ultimately a single global model is typically learned. As is usually the case for one-size-fits-all solutions, while the model may perform acceptably on average for many users, some clients may see very poor performance. Questions of fairness [24, 18] may arise if performance is compromised for individuals in the minority in favor of the majority.

Preprint. Under review.

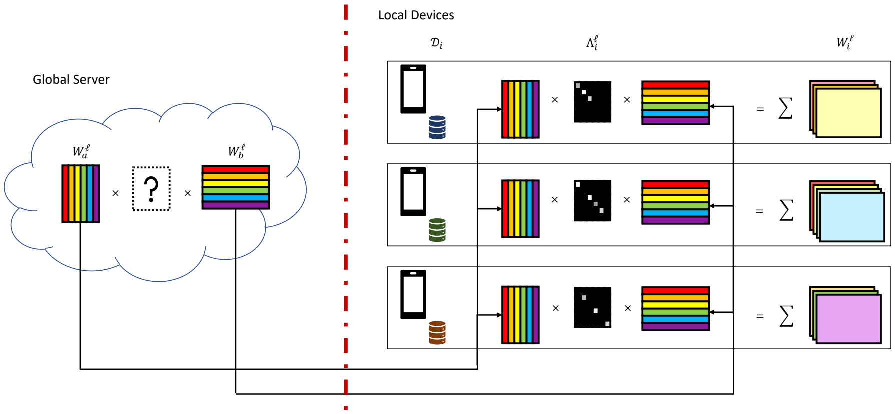


Figure 1: In WAFFLe, the clients share a global dictionary of rank-1 weight factors $\{W_a^\ell, W_b^\ell\}$. Each client uses a sparse diagonal matrix Λ_i^ℓ, specifying the combination of weight factors that constitute its own personalized model. Neither the client data $\mathcal{D}_i$ nor factor selections Λ_i^ℓ leave the local device.

Another ongoing challenge for federated learning is security. Data privacy is the primary motivation for keeping user data local on each device, rather than gathering it all in a centralized location for training. In more traditional distributed learning systems, data are exposed to additional vulnerabilities while being transmitted to and while residing in the central data repository. In lieu of the data, many federated learning approaches require clients to send weight updates to train the aggregated model. However, the threat of membership inference attacks [28], model inversion [7], or data leakage from gradients [37] mean that private data on each device can still be compromised if federated learning updates are intercepted or if the central server is breached.

We propose **W**eight **A**nonymized **F**actorization for **F**ederated **Le**arning (WAFFLe), leveraging Bayesian nonparametrics and neural network weight factorization to address these issues. Rather than learning a single global model, we learn a dictionary of rank-1 weight factor matrices. By selecting and weighting these factors, each local device can have a model customized to its unique data distribution, while sharing the learning burden of the weight factors across devices. We employ the Indian Buffet Process [8] as a prior to encourage factor sparsity and reuse of factors, performing variational inference to infer the distribution of factors for each client. While updates to the dictionary of factors are transmitted to the server, the distribution capturing which factors a client uses are kept local. This adds an extra insulating layer of security by obfuscating which factors a client is using, hindering an adversary's ability to perform membership inference attacks or dataset reconstruction.

To validate our approach, we perform experiments on MNIST [14], FMNIST [34], and CIFAR-10 [12] in settings exhibiting strong statistical heterogeneity. We observe that the model customization central to WAFFLe's design leads to higher performance for each client's local distribution, while also being more fair across all clients. Finally, we perform a membership inference attack [28] on WAFFLe, showing that it is more secure than FedAvg [21].

2 Methodology

2.1 Learning a Shared Dictionary of Weight Factors

Single Global Model Consider N client devices, with the i^{th} device having data distribution $\mathcal{D}_i$, which may differ as a function of i. In many distributed learning settings, a single global model is learned and deployed to all N clients. Thus, assuming a multilayer perceptron (MLP) architecture[1] with layers $\ell = 1, ..., L$, the set of weights $\theta = \{W^\ell\}_{\ell=1}^L$ is shared across all clients. To satisfy the global objective, θ is learned to minimize the loss on average across all clients. This is the approach of many federated learning approaches. For example, FedAvg [21] minimizes the following objective:

$$\min_\theta \mathscr{L}(\theta) = \sum_{i=1}^N p_i \mathcal{L}_i(\theta) \tag{1}$$

[1] While we restrict our discussion to fully connected layers here for simplicity, this can be generalized to other types of layers as well. See Appendix A for 2D *convolutional* layers.

where $\mathcal{L}_i(\theta) := \mathbb{E}_{x_i \sim \mathcal{D}_i}[l_i(x_i;\theta)]$ is the local objective function, N is the number of clients, and $p_i \geq 0$ is the weight of each device i. However, given statistical heterogeneity, such a one-size-fits-all approach may lead to the global model underfitting on certain clients; often this translates to how close a particular client's local distribution is to the population distribution. As a result, this model may be viewed as less fair to these clients with less common traits.

Individual Local Models On the other extreme, we may alternatively consider learning N local models $\theta_i = \{W_i^\ell\}_{\ell=1}^L$, each only trained on $\mathcal{D}_i$. In this case, each set of weights θ_i is maximally specific to the data distribution of each client i. However, each client typically has limited data, which may be insufficient for training a full model without overfitting; the total number of parameters that must be learned across all clients scales with N. Additionally, learning N separate models does not take advantage of similarities between client data distributions or the shared learning task.

Shared Weight Factors To make more efficient use of data, we instead propose a compromise between a single global model and N individual local models. Specifically, we allow each client's model to be personalized to the client's local distribution, but with all models sharing a dictionary of jointly learned components. Using a layer-wise decomposition [22], we construct each weight matrix with the following factorization:

$$W_i^\ell = W_a^\ell \Lambda_i^\ell W_b^\ell \tag{2}$$

$$\Lambda_i^\ell = \mathrm{diag}(\boldsymbol{\lambda}_i^\ell) \tag{3}$$

where $W_a^\ell \in \mathbb{R}^{J \times F}$ and $W_b^\ell \in \mathbb{R}^{F \times M}$ are global parameters shared across clients and $\boldsymbol{\lambda}_i^\ell \in \mathbb{R}^F$ is a client-specific vector. This factorization can be equivalently expressed as

$$W_i^\ell = \sum_{k=1}^{F} \lambda_{i,k}^\ell \left(\boldsymbol{w}_{a,k}^\ell \otimes \boldsymbol{w}_{b,k}^\ell \right) \tag{4}$$

where $\boldsymbol{w}_{a,k}^\ell$ is the k^{th} column of W_a^ℓ, $\boldsymbol{w}_{b,k}^\ell$ is the k^{th} row of W_b^ℓ, and $\otimes$ represents an outer product. Written in this way, the interpretation of the corresponding pairs of columns and rows $\boldsymbol{w}_{a,k}^\ell$ and $\boldsymbol{w}_{b,k}^\ell$ as weight *factors* is more apparent: W_a^ℓ and W_b^ℓ together comprise a global dictionary of the weight factors, and $\boldsymbol{\lambda}_i^\ell$ can be viewed as the factor *scores* of client i. Differences in $\boldsymbol{\lambda}_i^\ell$ between clients allows for customization of the model to each client's data distribution (see Figure 1), while sharing of the underlying factors W_a^ℓ and W_b^ℓ enables learning from the data of all clients.

We constitute each of the client's factor scores $\boldsymbol{\lambda}_i^\ell$ as the element-wise product:

$$\boldsymbol{\lambda}_i^\ell = \boldsymbol{r}^\ell \odot \boldsymbol{b}_i^\ell \tag{5}$$

where $\boldsymbol{r}^\ell \in \mathbb{R}^F$ indicates the strength of each factor and $\boldsymbol{b}_i^\ell \in \{0,1\}^F$ is a binary vector indicating the active factors. As explained below, $\boldsymbol{b}_i^\ell$ is typically sparse, so in general each client only uses a small subset of the available weight factors. Throughout this work, we use the absence of the ℓ superscript (*e.g.*, $\boldsymbol{\lambda}_i$) to refer to the entire collection across all layers for which this factorization is done. We learn a point-estimate for W_a, W_b and $\boldsymbol{r}$.

2.2 The Indian Buffet Process

Desiderata Within the context of federated learning with statistical heterogeneity, there are a number of desirable properties we wish the client factor scores to have collectively. Firstly, $\boldsymbol{\lambda}_i$ should be *sparse*, which encourages consolidation of related knowledge while minimizing interference: client A should be able to update the global factors during training without destroying client B's ability to perform its own task. This encourages *fairness*, as in settings with multiple subpopulations, this interference is most likely to be at the smaller groups' expense. On the other hand, we would also like factors to be reused among clients. While data may be non-i.i.d. across clients, there are often some similarities or overlap; thus, *shared* factors distribute learning across all clients' data, avoiding the N independent model's scenario. Finally, in the distributed settings considered in federated learning, the total number of nodes is rarely pre-defined. Therefore, there needs to be a way to gracefully *expand* to accommodate new clients to the system without re-initializing the whole model. This includes both increasing server-side capacity if necessary and initializing new clients.

Prior Given these desiderata, the Indian Buffet Process (IBP) [8] is a natural choice. As a prior, the IBP regularizes client factors to be sparse, and new factors are introduced but at a harmonic rate,

Algorithm 1 Weight Anonymized Factorization for Federated Learning (WAFFLe).

```
Input: Communication rounds $T$, local training epochs $E$, learning rate $\eta$
Server initializes global weight factor dictionaries $W_a$ and $W_b$, factor strengths $\boldsymbol{r}$
Clients each initialize variational parameters $\boldsymbol{\pi}_i, \boldsymbol{c}_i, \boldsymbol{d}_i$
for $t = 1, \cdots, T$ do
    Server randomly selects subset $\mathcal{S}_t$ of clients and sends $\{W_a, \boldsymbol{r}, W_b\}$
    for client $i \in \mathcal{S}_t$ in parallel do
        $W_a, \boldsymbol{r}, W_b, \boldsymbol{\pi}_i, \boldsymbol{c}_i, \boldsymbol{d}_i \leftarrow$ ClientUpdate($W_a, \boldsymbol{r}, W_b, \boldsymbol{\pi}_i, \boldsymbol{c}_i, \boldsymbol{d}_i$)
        Send $\{W_a, \boldsymbol{r}, W_b\}$ to the server.
    end for
    Server aggregates and averages updates $\{W_a, \boldsymbol{r}, W_b\}$
end for
function ClientUpdate($W_a, \boldsymbol{r}, W_b, \boldsymbol{\pi}_i, \boldsymbol{c}_i, \boldsymbol{d}_i$)
    for $e = 1, \cdots, E$ do
        for minibatch $b \in \mathcal{D}_i$ do
            Update $\{W_a, \boldsymbol{r}, W_b, \boldsymbol{\pi}_i, \boldsymbol{c}_i, \boldsymbol{d}_i\}$ by minimizing (12)
        end for
    end for
    Return $\{W_a, \boldsymbol{r}, W_b, \boldsymbol{\pi}_i, \boldsymbol{c}_i, \boldsymbol{d}_i\}$
end function
```

preferring reusing factors as much as possible over initializing new ones. This Bayesian nonparametric approach allows the data to dictate client factor assignment, factor reuse, and server-side model expansion. We use the stick-breaking construction of the IBP as a prior for factor selection:

$$v_{i,\kappa}^{\ell} \sim \mathrm{Beta}(\alpha, 1) \tag{6}$$

$$\pi_{i,k}^{\ell} = \prod_{\kappa=1}^{k} v_{i,\kappa}^{\ell} \tag{7}$$

$$b_{i,k}^{\ell} \sim \mathrm{Bernoulli}(\pi_{i,k}^{\ell}) \tag{8}$$

with α a hyperparameter controlling the expected number of active factors and the rate of new factors being incorporated, and k indexes the factor.

Inference We learn the posterior distribution for the random variables $\boldsymbol{\phi}_i = \{\boldsymbol{b}_i, \boldsymbol{v}_i\}$. Exact inference of the posterior is intractable, so we employ variational inference with mean-field approximation to determine the active factors for each client device, using the following variational distributions:

$$q(\boldsymbol{b}_i^{\ell}, \boldsymbol{v}_i^{\ell}) = q(\boldsymbol{b}_i^{\ell}) q(\boldsymbol{v}_i^{\ell}) \tag{9}$$

$$\boldsymbol{b}_i^{\ell} \sim \mathrm{Bernoulli}(\boldsymbol{\pi}_i^{\ell}) \tag{10}$$

$$\boldsymbol{v}_i^{\ell} \sim \mathrm{Kumaraswamy}(\boldsymbol{c}_i^{\ell}, \boldsymbol{d}_i^{\ell}) \tag{11}$$

learning the variational parameters $\{\boldsymbol{\pi}_i, \boldsymbol{c}_i, \boldsymbol{d}_i\}$ for each queried client using Bayes by Backprop [2]. Needing a differentiable parameterization, we use the Kumaraswamy distribution [13] as a replacement for the Beta distribution of $\boldsymbol{v}_i$ and utilize a soft relaxation of the Bernouilli distribution [20]. The objective for each client is to maximize the variational lower bound:

$$\mathcal{L}_i(\theta) = \sum_{n=1}^{|\mathcal{D}_i|} \mathbb{E}_q \log p\left(y_i^{(n)} | \boldsymbol{\phi}_i, x_i^{(n)}, W_a, W_b, \boldsymbol{r}\right) - \underbrace{\mathrm{KL}\left(q\left(\boldsymbol{\phi}_i\right) || p\left(\boldsymbol{\phi}_i\right)\right)}_{\mathscr{R}} \tag{12}$$

$$\mathscr{R} = \sum_{\ell=1}^{L} \mathrm{KL}\left(q(\boldsymbol{b}_i^{\ell}) || p(\boldsymbol{b}_i^{\ell} | \boldsymbol{v}_i^{\ell})\right) + \mathrm{KL}\left(q(\boldsymbol{v}_i^{\ell}) || p(\boldsymbol{v}_i^{\ell})\right) \tag{13}$$

where $\theta = \{W_a, W_b, \boldsymbol{r}, \boldsymbol{b}_i\}$ and $|\mathcal{D}_i|$ is the number of training examples at client i. Note that in (12) the first term provides label supervision and the second term ($\mathscr{R}$) regularizes the posterior not to stray far from the IBP prior.

2.3 Client-Server Communication

Training Before the training begins, the global weight factors $\{W_a, W_b\}$ and the factor strengths $\boldsymbol{r}$ are initialized by the server. Once initialized, each training round begins with $\{W_a, W_b, \boldsymbol{r}\}$ being sent to the selected subset of clients. Each sampled client then trains the model on their own private dataset $\mathcal{D}_i$ for E epochs, updating not only the weight factor dictionary $\{W_a, W_b\}$ and the factor strengths r, but also its also own variational parameters $\{\boldsymbol{\pi}_i, \boldsymbol{c}_i, \boldsymbol{d}_i\}$, which controls which factors it uses. Once local training is finished, each client sends $\{W_a, W_b, \boldsymbol{r}\}$ back to the server, but not $\{\boldsymbol{\pi}_i, \boldsymbol{c}_i, \boldsymbol{d}_i\}$, which remain with the client with data $\mathcal{D}_i$. After the server has received back updates from all clients, the various new values for $\{W_a, W_b, \boldsymbol{r}\}$ are aggregated with a simple averaging step. The process then repeats, with the server selecting a new subset of clients to query, sending the new updated set of global parameters, until the desired number of communication rounds have passed. This process is summarized in Algorithm 1.

Evaluation When a client enters the evaluation mode, it requests the current version of global parameters $\{W_a, W_b, \boldsymbol{r}\}$ from the server. If the client has been previously queried for federated training, the local model consists of the aggregated global parameters and the binary vector generated by its own local variational parameters $\{\boldsymbol{\pi}_i\}$. Otherwise, the client uses only the aggregated $\{W_a, W_b, \boldsymbol{r}\}$. Note that if a client has been previously queried, the most recently cached copy of the global parameters is an option if a network connection is unavailable or too expensive; in our experiments, we assume clients are able to request the most up-to-date parameters.

Security Data security is one of the central tenets of federated learning. Simpler, more standard methods of training a model could be utilized if all data were first aggregated at a central server. However, sensitive client data being intercepted during transmission or the server's data repository being breached by an attacker are major concerns, motivating federated learning's approach of keeping the data on the local device. On the other hand, keeping the data client-side may not be sufficient. Just as data can be compromised in transit or at the central database in non-federated settings, federated training updates are similarly vulnerable. In methods like FedAvg, this update is the entirety of the model's parameters. Effectively, this means that FedAvg trades yielding the data immediately for surrendering whitebox access to the model, which opens the model to a wide range of malicious activities [31, 7, 28, 37], including, critically, exposing the very data that federated learning aims to protect. With WAFFLe, clients transmit back the entire dictionary of weight factors $\{W_a, W_b\}$ and $\boldsymbol{r}$, but not $\{\boldsymbol{\pi}_i, \boldsymbol{c}_i, \boldsymbol{d}_i\}$. As such, the knowledge of which specific factors that a particular client uses is kept local. Therefore, even if messages are intercepted, an adversary cannot completely reconstruct the model, hampering their ability to perform attacks to recover the data.

3 Related Work

3.1 Statistical Heterogeneity

Statistical heterogeneity of the data distributions of client devices has long been recognized as a challenge for federated learning. Despite acknowledging statistical heterogeneity, many federated learning algorithms still focus on learning a single global model [21]; such an approach often suffers from divergence of the model, as local models may vary significantly from each other. To address this challenge, a number of works break away from the single-global-model formulation. Several [29, 3] have cast federated learning as a multi-task learning problem, with each client treated as a separate task. FedProx [17] adds a proximal term to account for statistical heterogeneity by limiting the impact of local updates. Others study federated learning within a model-agnostic meta-learning framework [10, 11]. In [36] performance degradation from non-i.i.d. data is recognized, proposing global sharing of a small subset of data, which while effective, may compromise privacy. In settings of high statistical heterogeneity, fairness is also a natural question. AFL [24] and q-FFL [18] both propose methods of focusing the optimization objective on the clients with the worst performance, though they do not change the network itself to model different data distributions.

3.2 Preserving Privacy

While much progress has been made in machine learning with public datasets [14, 12, 5], in real-world settings, data are often more sensitive, potentially for propriety [30], security [19], or privacy [27] reasons. This concern for the data is one of the primary motivations for federated learning in the first place. Previous methods have focused on differential privacy [6], such as adding noise to the learned model parameters [1, 23]. However, in practice, sharing the whole model architecture and all its parameters still presents risks associated with whitebox access, leaving the data vulnerable to attacks such as membership inference attack [28], model inversion [7], and deep leakage [37].

3.3 Bayesian Nonparametric Federated Learning

Several previous works have applied Bayesian nonparameterics to federated learning, though primarily as a means for parameter matching during aggregation. Instead of averaging the parameters weight-wise without considering the meaning of each parameter, past works [35, 33] have proposed using the Beta-Bernouilli Process [32] for matching parameters. Specifically, FedMA [33] is an improved version of PFNM [35] that extends the matching from fully connected layers to convolution and LSTM units [9]. However, the matching process requires k-means clustering with additional constraints on the assignment, which must be permutation matrices. In contrast, our method utilizes Bayesian nonparametrics for modeling rank-1 factors for multitask learning, instead of the aggregation stage.

4 Experiments

4.1 Experimental Set-up

Settings with higher statistical heterogeneity are more challenging for federated learning than when data are i.i.d. across clients, as well as more representative of the real-world, so we focus our experiments on the former. We consider two forms of statistical heterogeneity. The first is the simple non-i.i.d. construction introduced by [21], in which the data are sorted by class, sharded, and then randomly distributed to the N clients such that each client only has data from Z classes; many federated learning works consider the highly non-i.i.d. setting of $Z = 2$, which we also default to. While this setting can be challenging, it has the property that the classes present in every client's data is equally represented in the global data distribution. As a result, a single global model may perform reasonably uniformly across all clients. We thus refer to this as *unimodal* non-i.i.d.

However, this assumption of equal representation is generally not true in practice, as some characteristics or modes of the global distribution are inevitably less prevalent in the overall population than others. In the real world, this can correspond to age, gender, ethnicity, wealth, or a number of other demographic factors. To emulate this, we modify the above non-i.i.d. setting by first splitting the data and clients into two groups, with more clients in one group than the other. For MNIST [14] for example, we partition the odd digits to 100 clients and the even digits to 20 clients. As before, each client still receives data from $Z = 2$ classes, with an equal number of data samples per client (the unallocated even digit samples are left unused); the difference is that there is now a $5 : 1$ ratio of odd to even digits in the total population, resulting in the clients with only even digits being in the minority of the global population. We call this setting *multimodal* non-i.i.d. Further details on the data allocation process and splits for FMNIST [34] and CIFAR-10 [12] can be found in Appendix B.

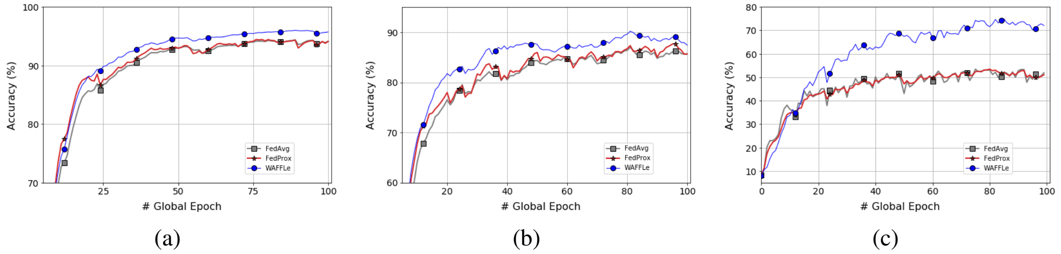
In our experiments, the server selects a fraction $C = 0.1$ of clients during each communication round, with $T = 100$ total rounds for all methods. Each selected client trains their own model for $E = 5$ local epochs with mini-batch size $B = 10$, and the FedProx [17] proximal parameter μ is set to 1.0. Model architectures, settings of F and α, and training schedules for each of the datasets are described in Appendix C. Ablation studies over the number of local epochs E, number of classes per client Z, and IBP parameters α and F are provided in Appendix D, demonstrating robustness.

4.2 Local Test Performance

We compare WAFFLe with FedAvg [21] and FedProx [17], which augments FedAvg with a proximal term designed for settings with high statistical heterogeneity. We plot in Figures 2 and 3 local test performance averaged across all clients over time for both types of non-i.i.d. data allocation, and we

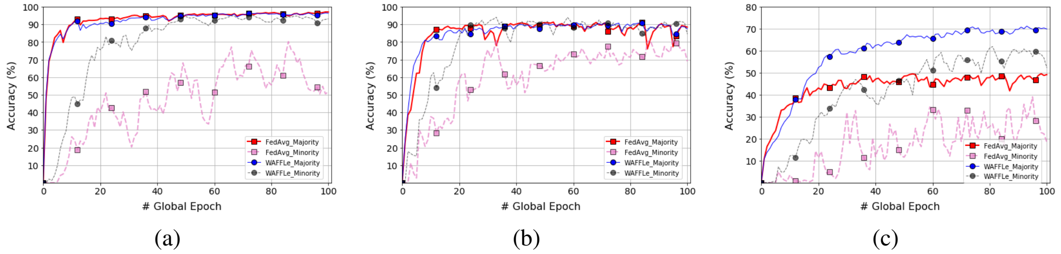
Table 1: Local Test Performance for $Z = 2$

Dataset	Method	# of parameters ↓	Unimodal ↑	Multimodal ↑
MNIST	FedAvg	155,800	94.46±0.84	91.57±1.42
	FedProx	155,800	94.44±1.15	91.53±1.05
	WAFFLe	**120,200**	**96.23**±0.31	**95.41**±0.36
FMNIST	FedAvg	28,880	83.96±0.91	83.43±2.27
	FedProx	28,800	84.19±0.99	83.59±2.30
	WAFFLe	**18,155**	**87.12**±0.89	**86.09**±0.92
CIFAR-10	FedAvg	61,770	52.54±0.14	45.46±1.69
	FedProx	61,770	52.36±0.11	44.95±1.17
	WAFFLe	**42,780**	**71.30**±0.92	**66.35**±0.72



(a) (b) (c)

Figure 2: Local test performance for unimodal non-*i.i.d.* degree $Z = 2$. (a) MNIST; (b) FMNIST; (c) CIFAR-10.



(a) (b) (c)

Figure 3: Local test performance for majority and minority subpopulations for multimodal non-*i.i.d.* degree $Z = 2$. (a) MNIST; (b) FMNIST; (c) CIFAR-10.

record final local test performance in Table 1, along with the total number of learnable parameters. WAFFLe performs well despite strong statistical heterogeneity, as each client can learn a personalized model by selecting different factors from $\{W_a, W_b\}$; having a model specific to each data distribution results in higher local test performance than the baselines. This advantage over other methods is especially apparent when the data are distributed multimodal non-i.i.d., mainly because WAFFLe more effectively models underrepresented clients.

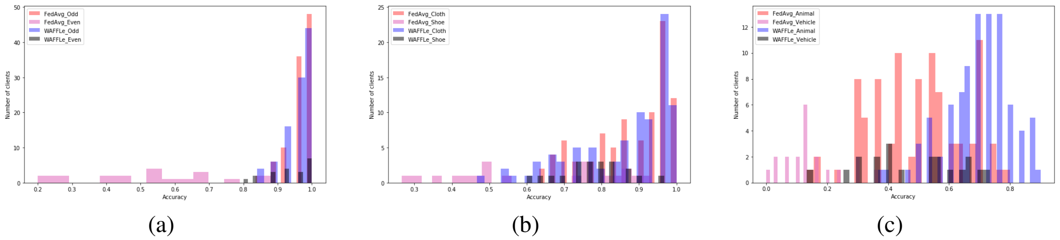
Interestingly, we find that WAFFLe outperforms the baselines particularly significantly for CIFAR-10, the most challenging of the tested datasets, with WAFFLe's local test performance outstripping the other methods by 18.8% and 20.9% for unimodal and multimodal settings, respectively. This demonstrates WAFFLe's ability to scale to complex tasks beyond MNIST, a common federated learning test bed. Additionally, note that even though WAFFLe effectively learns a different model for each client, this does not lead to the computation or memory costs typically associated with independent models. By sharing rank-1 factors, each weight factor is represented compactly, resulting in a total number of parameters that is *fewer* that the single model used by FedAvg and FedProx, despite using the same architecture.

Table 2: Sub-population Local Test Performance Analysis

Dataset	Method	Majority ↑	Minority ↑	Gap ↓	Variance ↓
MNIST	FedAvg	**96.63**±0.70	67.40±11.26	29.23±11.79	199±106
	FedProx	96.43±0.67	68.60±9.44	27.83±10.03	186±92
	WAFFLe	95.93±0.16	**93.87**±0.66	**2.07**±0.77	**26**±6
FMNIST	FedAvg	89.75±1.76	68.05±4.43	21.70±4.21	231±35
	FedProx	**89.95**±1.73	67.50±4.50	22.45±4.38	233±42
	WAFFLe	88.91±2.07	**79.67**±1.52	**9.25**±0.61	**145**±27
CIFAR-10	FedAvg	51.98±1.69	16.83±4.42	35.15±4.12	338±59
	FedProx	51.26±1.44	16.56±3.32	32.70±6.99	318±36
	WAFFLe	**68.37**±1.01	**55.00**±6.00	**13.37**±2.61	**182**±27

4.3 Fairness

We visualize in Figure 4 the distribution of final local test performance of FedAvg and WAFFLe for each client in the majority and minority groups for all three datasets, summarizing subpopulation mean performance and overall population variance in Table 2. We observe that FedAvg, which learns



(a) (b) (c)

Figure 4: Performance distribution across clients in the multimodal non-i.i.d. setting for (a) MNIST, (b) FMNIST and (c) CIFAR-10.

a single global model, focuses on minimizing mean error across the population, resulting in stronger performance for the clients in the majority. However, as a result, clients in the minority are severely compromised, as evidenced by the large difference ("Gap") between majority and minority values in Table 2; for example, FedAvg's performance for the "evens" group of clients is almost 30% lower than that of the "odds" group. This underfitting can be seen to exist throughout training from the "FedAvg_Minority" curve in Figure 3, which lags far below the "FedAvg_Majority" in all three datasets. On the other hand, because of WAFFLe's shared weight factor dictionary design, different knowledge can be encoded in separate weight factors, which can be used by different parts of the population. As a result, despite certain classes being underrepresented (both in terms of clients, and total samples) in the training set, WAFFLe is able to successfully model them, with performances on par with the overall population. Notably, we achieve this without explicitly enforcing fairness through client sampling during training [24, 18], which can be incorporated to further encourage uniform performance across clients.

4.4 Membership Inference Attack

A primary purpose of federated learning is to keep data safe. However, as mentioned in Section 2.3, the predominant federated learning strategy of each client sending their entire updated model's weights still leaves the client's data vulnerable. Given a data query, membership inference attacks (MIAs) [28] can be used to infer if it was used during model training, leveraging the tendency of machine learning to overfit or memorize training data. As such, a successful MIA can be used by an attacker to surmise the content of a client's private data from the model. We demonstrate this by performing a MIA in a federated setting.

We begin with a LeNet [14] FedAvg [21] model trained on CIFAR-10, with each client having 1000 training examples. Following [28], we then train a small ensemble of 3 "shadow" models and then use them for MIA. As shown in Table 3, this simple attack achieves a high success rate at identifying a FedAvg client's training data, as intercepting the training update gives the full model. On the other hand, WAFFLe's training update transmissions only send partial model information, as the identity of the active factors is kept private. As a result, MIA success rate on WAFFLe is only moderately higher than random chance (50%). This means it is significantly harder to identify the private training data for WAFFLe, relative to FedAvg.

Table 3: Membership Inference Attacks

Methods	Accuracy	F1-score
FedAvg	83.85± 1.62	83.72 ± 2.19
WAFFLe	**56.20** ± 1.40	**54.39** ± 1.85

5 Conclusion

We have introduced WAFFLe, a Bayesian nonparametric framework using shared rank-1 weight factors for federated learning. This approach allows for learning individual models for each client's specific data distributions while still sharing the underlying learning problem in a parameter-efficient manner. Our experiments demonstrate that this model customizability makes WAFFLe successful at improving local test performance and, more importantly, significantly improves fairness in model performance when the data distribution among clients is multimodal. Furthermore, we are able to scale our results to CIFAR-10 and convolutional networks, where we actually observe the biggest improvements. We also show that by keeping the active factors selected by each model private on each device along with the data, WAFFLe's communication rounds only send partial model information, making it significantly harder to perform attacks on the private data.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep Learning with Differential Privacy. *ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[2] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Networks. *International Conference on Machine Learning*, 2015.

[3] Luca Corinzia and Joachim M Buhmann. Variational Federated Multi-Task Learning. *arXiv preprint arXiv:1906.06268*, 2019.

[4] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. Large Scale Distributed Deep Networks. *Neural Information Processing Systems*, 2012.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. ImageNet: A Large-scale Hierarchical Image Database. *Computer Vision and Pattern Recognition*, 2009.

[6] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating Noise to Sensitivity in Private Data Analysis. *Theory of Cryptography Conference*, pages 265–284, 2006.

[7] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. *ACM SIGSAC Conference on Computer and Communications Security*, 2015.

[8] Zoubin Ghahramani and Thomas L Griffiths. Infinite Latent Feature Models and the Indian Buffet Process. *Neural Information Processing Systems*, 2006.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 1997.

[10] Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. Improving Federated Learning Personalization via Model Agnostic Meta Learning. *arXiv preprint arXiv:1909.12488*, 2019.

[11] Mikhail Khodak, Maria-Florina F Balcan, and Ameet S Talwalkar. Adaptive Gradient-based Meta-learning Methods. *Neural Information Processing Systems*, 2019.

[12] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. 2009.

[13] Ponnambalam Kumaraswamy. A Generalized Probability Density Function for Double-Bounded Random Processes. *Journal of Hydrology*, 1980.

[14] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[15] Daliang Li and Junpu Wang. FedMD: Heterogenous Federated Learning via Model Distillation. *arXiv preprint arXiv:1910.03581*, 2019.

[16] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated Learning: Challenges, Methods, and Future Directions. *arXiv preprint arXiv:1908.07873*, 2019.

[17] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated Optimization in Heterogeneous Networks. *arXiv preprint arXiv:1812.06127*, 2018.

[18] Tian Li, Maziar Sanjabi, and Virginia Smith. Fair Resource Allocation in Federated Learning. *International Conference on Learning Representations*, 2020.

[19] Kevin J Liang, Geert Heilmann, Christopher Gregory, Souleymane O. Diallo, David Carlson, Gregory P. Spell, John B. Sigman, Kris Roe, and Lawrence Carin. Automatic Threat Recognition of Prohibited Items at Aviation Checkpoints with X-ray Imaging: A Deep Learning Approach. *SPIE Anomaly Detection and Imaging with X-Rays (ADIX) III*, 2018.

[20] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. *International Conference on Learning Representations*, 2017.

[21] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient Learning of Deep Networks from Decentralized Data. *Artificial Intelligence and Statistics*, 2017.

[22] Nikhil Mehta, Kevin J Liang, and Lawrence Carin. Bayesian Nonparametric Weight Factorization for Continual Learning. *arXiv preprint arXiv:2004.10098*, 2020.

[23] Luca Melis, George Danezis, and Emiliano De Cristofaro. Efficient Private Statistics with Succinct Sketches. *arXiv preprint arXiv:1508.06110*, 2015.

[24] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic Federated Learning. *International Conference on Machine Learning*, 2019.

[25] Vinod Nair and Geoffrey E Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. *International Conference on Machine Learning*, 2010.

[26] Daniel Peterson, Pallika Kanani, and Virendra J Marathe. Private Federated Learning with Domain Adaptation. *arXiv preprint arXiv:1912.06733*, 2019.

[27] Dezsö Ribli, Anna Horváth, Zsuzsa Unger, Péter Pollner, and István Csabai. Detecting and Classifying Lesions in Mammograms with Deep Learning. *Nature Scientific Reports*, 8, 2018.

[28] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks Against Machine Learning Models. *IEEE Symposium on Security and Privacy (SP)*, 2017.

[29] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated Multi-task Learning. *Neural Information Processing Systems*, 2017.

[30] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. *International Conference on Computer Vision*, 2017.

[31] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing Properties of Neural Networks. *International Conference on Learning Representations*, 2014.

[32] Romain Thibaux and Michael I Jordan. Hierarchical Beta Processes and the Indian Buffet Process. *Artificial Intelligence and Statistics*, 2007.

[33] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated Learning with Matched Averaging. *arXiv preprint arXiv:2002.06440*, 2020.

[34] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[35] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Trong Nghia Hoang, and Yasaman Khazaeni. Bayesian Nonparametric Federated Learning of Neural Networks. *arXiv preprint arXiv:1905.12022*, 2019.

[36] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated Learning with Non-IID Data. *arXiv preprint arXiv:1806.00582*, 2018.

[37] Ligeng Zhu, Zhijian Liu, and Song Han. Deep Leakage from Gradients. *Neural Information Processing Systems*, 2019.

A Generalizing Weight Factorization to Convolutional Kernels

While introducing WAFFLe's formulation in Section 2.1, we assumed a multilayer perceptron (MLP) model, as illustrating our proposed shared dictionary with the 2D weight matrices composing fully connected layers is made especially clearer. While MLPs are sufficient for simple datasets such as MNIST, more challenging datasets require more complex architectures to achieve the most competitive results. For computer vision, for example, this often means convolutional layers, whose kernels are 4D. While 4D tensors can be similarly decomposed into rank-1 factors with tensor rank decomposition, such an approach would result in a large increase in the number of parameters in the weight factor dictionary due to the low spatial dimensions of the convolutional kernels (*e.g.*, 3×3) in most commonly used architectures. Instead, we reshape the 4D convolutional kernels into 2D matrices by combining the three input dimensions (number of input channels, kernel width, and kernel height) into a single input dimension. We then proceed with the formulation in (2). Similar approaches can be taken to generalize our formulation to other types of layers.

B Data Partitioning

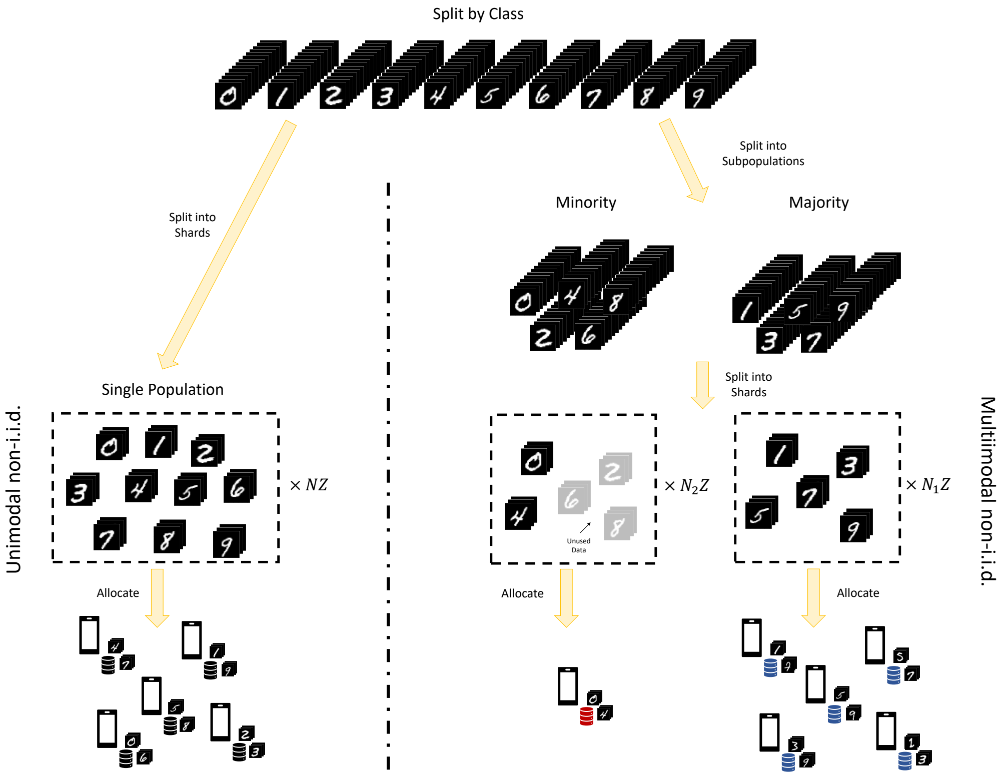


Figure 5: Example data allocation process to N clients for MNIST and $Z = 2$ in the unimodal i.i.d. (left) and multimodal i.i.d. (right) settings. Notice that the primary difference is the grouping of the data into two subpopulations (here referred to as "Majority" and "Minority") before sharding and allocating Z shards to each client.

Because statistical heterogeneity is an inherent property of federated learning paradigms, we focus our evaluation in this setting, testing WAFFLe in two different types of non-i.i.d. partitions of the data. A diagram showing the differences of the data allocation process for the two considered settings is shown for MNIST [14] in Figure 5.

B.1 Unimodal non-i.i.d.

We first consider the non-i.i.d. setting introduced by [21]. This is a widely used evaluation setting, commonly referred to as "non-i.i.d." or "heterogeneous" in other federated learning works, to distinguish it from completely i.i.d. data splits. We refer to this as *unimodal non-i.i.d.* to distinguish it from our second setting, which is also non-i.i.d. The primary purpose of such a partition is to investigate the behavior of federated average algorithms when each client has data from only a subset (Z) of classes.

This type of partition begins by sorting all data by class. Given N client devices, the samples from each class are evenly divided into shards of data, each consisting of a single class, resulting in NZ shards across all classes. These shards are then randomly distributed to the N clients such that each receives Z shards. The data in the Z shards for each client is then shuffled together and split into a local training and test set. This ensures that the local test set for each client is representative of its own private data distribution.

B.2 Multimodal non-i.i.d.

While the above partition does explore the non-i.i.d. nature of class distribution among clients, it does not adequately characterize the tendency for subpopulations to exist, with some being more prevalent than others. We propose a new non-i.i.d. setting to capture this, which we call *multimodal non-i.i.d.*, as each subpopulation group can be thought of as a mode of the overall distribution.

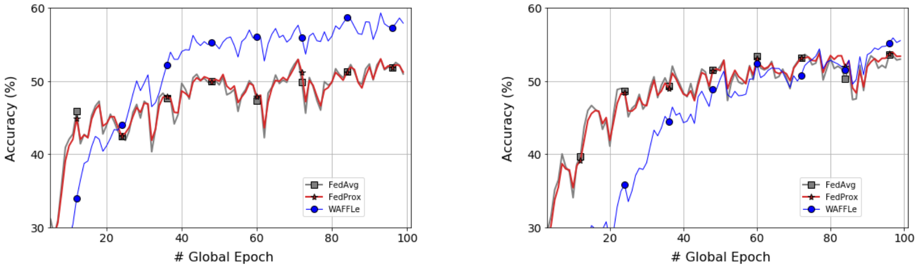
This partition begins similarly to unimodal non-i.i.d., with the data being sorted by class. Before sharding, however, classes are assigned to modes. The number of modes is arbitrary, but we choose two for simplicity, creating “majority” and “minority” subpopulations. In our experiments, the two modes are odd digits ($N_1 = 100$) versus even digits ($N_2 = 20$) for MNIST [14], footwear and shirts ($N_2 = 20$) versus everything else ($N_1 = 90$) for FMNIST [34], and animals ($N_1 = 90$) versus vehicles ($N_2 = 20$) for CIFAR-10 [12], where N_1 and N_2 are the number of clients in the majority and minority subpopulations, respectively. Once the classes have been separated by group, the process proceeds similarly to the unimodal i.i.d. partition process, with the data being divided into shards and then randomly allocated to clients within each subpopulation. We make the shards equal in size both within and across modes, so in instances where there are more data shards available than there are clients, we discard the unallocated data. Just as for unimodal non-i.i.d., local training and test sets are created for each client from its allocated data.

C Additional Experimental Set-up Details

MNIST For MNIST [14] digit recognition, we use a multilayer perceptron with 1-hidden layer with 200 units using ReLU activations [25]. Based on this model, we constructed WAFFLe with $F = 120$ factors. The traditional 60K training examples are partitioned into local training and test sets as described in Section 4.1. Stochastic gradient descent (SGD) with learning rate $\eta = 0.04$ is employed for all methods.

FMNIST For FMNIST [34] fashion recognition, we use a convolutional network consisting of two 5×5 convolution layers with 16 and 32 output channels respectively. Each convolution layer is followed by a 2×2 maxpooling operation with ReLU activations. A fully connected layer with a softmax is added for the output. Based on this model, we construct WAFFLe by only factorizing the convolution layers, with $F = 25$ factors. As with MNIST, the traditional 60K training examples are used to form the two local sets. SGD with learning rate $\eta = 0.02$ is used as the optimizer for all methods.

CIFAR-10 For CIFAR-10 [12], we use we use a convolutional network consisting of two 3×3 convolution layers with 16 and 16 output channels respectively. Each convolution layer is followed by a 2×2 maxpooling operation with ReLU activations. These two convolutions are followed by two fully-connected layers with hidden size 80 and 60, with a softmax applied for the final output probabilities. To construct WAFFLe, we set the number of factors $F = 10$ for the two convolution layers, $F = 80$ for the first fully connected layer, and $F = 40$ for the second fully connected layer. The 50K training examples are used for constructing the local train and test sets. SGD with learning rate $\eta = 0.02$ is utilized for all methods.



(a)

(b)

Figure 6: CIFAR-10 local test performance for statistical heterogeneity: (a) $Z = 3$; (b) $Z = 4$.

Table 4: Unimodal Local Test Accuracy vs Local Epochs

Dataset	Method	E=10	E=20	E=30
MNIST	FedAvg WAFFLe	92.95 95.10	93.36 96.32	93.55 96.43
FMNIST	FedAvg WAFFLe	85.32 87.52	85.13 87.07	85.14 89.25
CIFAR-10	FedAvg WAFFLe	47.40 64.18	47.60 71.92	55.39 74.50

Table 5: Multimodal Local Test Accuracy vs Local Epochs

Dataset	Method	E=10	E=20	E=30
MNIST	FedAvg WAFFLe	88.70 95.37	89.27 94.87	89.03 95.07
FMNIST	FedAvg WAFFLe	86.21 87.03	86.58 89.15	86.47 91.33
CIFAR-10	FedAvg WAFFLe	40.91 58.79	42.09 57.00	42.00 62.61

D Ablation Studies

Statistical Heterogeneity (Z) WAFFLe is specifically designed for statistical heterogeneity, as each client can select different weight factors, effectively learning personalized models. WAFFLe was shown to excel when $Z = 2$, as this is a strongly non-i.i.d. setting: as each client only has samples from two classes. In Figure 6, we show how WAFFLe performs in unimodal settings with less statistical heterogeneity, for $Z = \{3, 4\}$. Although it takes longer to converge in these cases, WAFFLe still outperforms FedAvg by 7.20% and 2.74%, respectively.

Table 6: Unimodal Local Test Accuracy vs α and F

	F=80	F=100	F=150
α/F = 0.4	93.20	94.07	94.42
α/F = 0.6	95.08	94.48	95.56
α/F = 0.8	95.56	95.15	96.08
α/F = 1.0	96.33	95.63	96.45

Table 7: Multimodal Local Test Accuracy vs α and F

	F=80	F=100	F=150
α/F = 0.4	91.83	92.70	93.23
α/F = 0.6	94.23	94.48	95.26
α/F = 0.8	94.76	95.15	95.70
α/F = 1.0	94.70	94.93	95.93

Local epochs (E) Training client devices for more local epochs allows each server to collect a bigger update from each device, increasing local computation in exchange for fewer total communication rounds. This is often a desirable trade-off, as communication costs are commonly viewed as the primary bottleneck for federated learning. However, too many local epochs can lead to divergence during the aggregation step. We study the influence of local epochs E for unimodal non-i.i.d. in Table 4 and for multimodal non-i.i.d. in Table 5, using the same settings as in Section 4.1 except for reducing the global training epochs T to 50 and the learning rate η to 0.02 for all methods in multimodal non-i.i.d scenario. We observe that WAFFLe can handle increased number of local epochs, improving performance for all three datasets.

Indian Buffet Process Sparsity (α) and Number of Factors (F) At the cost of more parameters, an increasing number factors F and higher IBP parameter α gives client more expressivity for modeling its local distribution.

We study the influence of α and F for an MLP architecture on MNIST partitioned in both non-i.i.d. settings in Tables 6 and 7. As expected, the higher α and F are, the better performance we observe, though in practice we prefer lower α and F for efficiency. On the other hand, the overall difference in local test accuracy does not vary drastically, meaning that WAFFLe is fairly robust to both hyperparameters.