IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 18, 2022, accepted April 29, 2022, date of publication May 5, 2022, date of current version May 12, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3172945

WAFFLe: Weight Anonymized Factorization for Federated Learning

WEITUO HAO[1], NIKHIL MEHTA[1], KEVIN J. LIANG[1], PENGYU CHENG[1], MOSTAFA EL-KHAMY[2,3], (Senior Member, IEEE), AND LAWRENCE CARIN[4], (Fellow, IEEE)

[1]Duke University, Durham, NC 27708, USA
[2]SOC Research and Development, Samsung Semiconductor Incorporation (SSI), San Diego, CA 92121, USA
[3]Faculty of Engineering, Alexandria University, Alexandria 21544, Egypt
[4]King Abdullah University of Science and Technology, Thuwal 23955, Saudi Arabia

Corresponding author: Weituo Hao (weituo.hao@duke.edu)

This work was supported in part by the SOC Research and Development of Samsung Semiconductor Incorporation, USA.

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.

INDEX TERMS Federated learning, Indian buffet process, personalization and fairness.

I. 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 [1] 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 remain for federated learning [2].

As with centralized distributed learning settings [3], many federated learning algorithms focus on learning a single global model. However, due to variation in user characteristics, personal data are likely to exhibit significant *statistical heterogeneity*. To simulate this, federated learning algorithms are commonly tested in non-i.i.d. settings [1], [4]–[6], 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 poor performance. Questions of fairness [7], [8] may arise if performance is compromised for individuals in the minority in favor of the majority.

Another challenge for federated learning is security. Data privacy is the primary motivation for keeping user data local on each device, rather than gathering it in a centralized

The associate editor coordinating the review of this manuscript and approving it for publication was Chao Tong.

location for training. In 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 [9], [10] or model inversion [11], [12] 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. We make the following contributions: *i*) 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. *ii*) We employ the Indian Buffet Process [13] as a prior to encourage factor sparsity and reuse of factors, performing variational inference to infer the distribution of factors for each client. *iii*) 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. *iv*) Finally, individually customized models represent in more fairness.

We perform experiments on MNIST [14], FMNIST [15], and CIFAR-10 [16] 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 significantly fairer across all clients. Finally, we perform membership inference [9] and model inversion [11] attacks on WAFFLe, showing that it is much harder to expose user data than with FedAvg [1].

II. METHODOLOGY

A. SHARED DICTIONARY OF WEIGHT FACTORS

1) 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 $\ell = 1, \ldots, L$, 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 [1] minimizes the following objective:

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

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.

2) 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 leverage similarities between client data distributions or the shared learning task.

3) 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 [17], we construct each weight matrix with the following factorization:

$$W_i^\ell = W_a^\ell \Lambda_i^\ell W_b^\ell, \quad \Lambda_i^\ell = \text{diag}(\boldsymbol{\lambda}_i^\ell) \tag{2}$$

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. Note that the construction is not a post-training processing such as singular value decomposition(SVD) on trained weights but a parameters format before training. This factorization can be equivalently expressed as

$$W_i^\ell = \sum_{k=1}^{F} \lambda_{i,k}^\ell \left(\boldsymbol{w}_{a,k}^\ell \otimes \boldsymbol{w}_{b,k}^\ell \right) \tag{3}$$

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 used to select the corresponding rank-1 matrices formed using weight factors. 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.

[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.

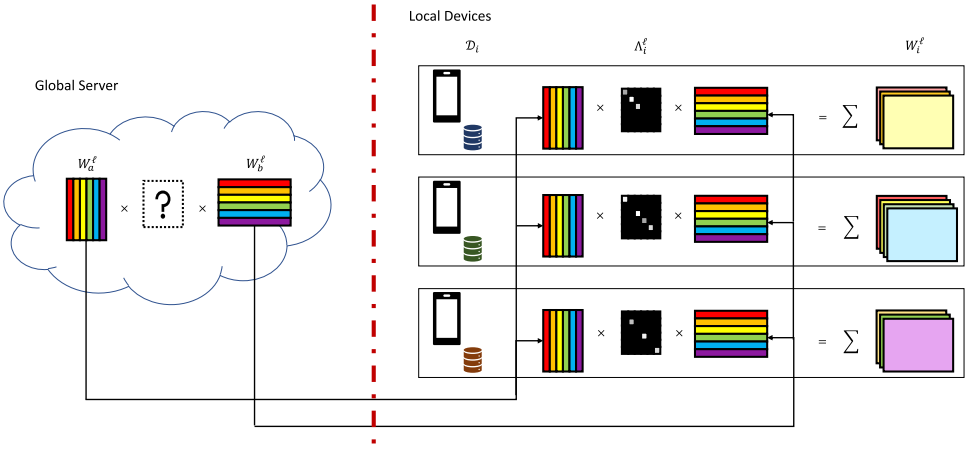


FIGURE 1. In WAFFLe, the clients share a global dictionary of rank-1 weight factors $\{W_a^\ell, W_b^\ell\}$. Each client uses a sparse diagonal matrix Λ_i^ℓ, specifying the combination of weight factors that constitute its own personalized model. Neither the client data $\mathcal{D}_i$ nor factor selections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

Algorithm 1 Updating Scheme in Each Communication

Input: local training epochs E, learning rate η
Server randomly selects subset $\mathcal{S}_t$ of clients
Server sends $\{W_a, \boldsymbol{r}, W_b\}$ to $\mathcal{S}_t$
for client $i \in \mathcal{S}_t$ **in parallel do**
  $W_a, \boldsymbol{r}, W_b, \boldsymbol{\pi}_i \boldsymbol{c}_i, \boldsymbol{d}_i \leftarrow$ CLIENTSTEP$(W_a, \boldsymbol{r}, W_b, \boldsymbol{\pi}_i, \boldsymbol{c}_i, \boldsymbol{d}_i)$
  Send $\{W_a, \boldsymbol{r}, W_b\}$ to the server.
end for
Server aggregates and averages updates $\{W_a, \boldsymbol{r}, W_b\}$

Algorithm 2 Client Updating Function

function CLIENTSTEP$(W_a, \boldsymbol{r}, W_b, \boldsymbol{\pi}_i, \boldsymbol{c}_i, \boldsymbol{d}_i)$
  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 (11)
    end for
  end for
  return $W_a, \boldsymbol{r}, W_b, \boldsymbol{\pi}_i, \boldsymbol{c}_i, \boldsymbol{d}_i$
end function

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

$$\boldsymbol{\lambda}_i^\ell = \boldsymbol{r}^\ell \odot \boldsymbol{b}_i^\ell \tag{4}$$

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 $\boldsymbol{r}$.

B. THE INDIAN BUFFET PROCESS

1) 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; 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.

2) PRIOR

Given these desiderata, the Indian Buffet Process (IBP) [13] 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, 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 [18] as a prior for the factor selection:

$$v_{i,\kappa}^\ell \sim \text{Beta}(\alpha, 1) \tag{5}$$

$$\pi_{i,k}^\ell = \prod_{\kappa=1}^{k} v_{i,\kappa}^\ell \tag{6}$$

$$b_{i,k}^\ell \sim \text{Bernoulli}(\pi_{i,k}^\ell) \tag{7}$$

where k indexes the factor, $\pi_{i,k}^{\ell}$ denotes the probability of the k^{th} factor being active, and α is a hyperparameter controlling the expected number of active factors and the rate of new factors being incorporated. Note that in the stick-breaking construction, $\pi_{i,k}^{\ell}$ is generated using a cumulative product of Beta random variables ($v_{i,\kappa}^{\ell}$).

3) 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 variational distributions:

$$q(\boldsymbol{b}_i^{\ell}, \boldsymbol{v}_i^{\ell}) = q(\boldsymbol{b}_i^{\ell}) q(\boldsymbol{v}_i^{\ell}) \tag{8}$$

$$\boldsymbol{b}_i^{\ell} \sim \text{Bernoulli}(\boldsymbol{\pi}_i^{\ell}) \tag{9}$$

$$\boldsymbol{v}_i^{\ell} \sim \text{Kumaraswamy}(\boldsymbol{c}_i^{\ell}, \boldsymbol{d}_i^{\ell}) \tag{10}$$

learning the variational parameters $\{\boldsymbol{\pi}_i, \boldsymbol{c}_i, \boldsymbol{d}_i\}$ for each queried client using Bayes by Backprop [19]. Needing a differentiable parameterization, we use the Kumaraswamy distribution [20] as a replacement for the Beta distribution of $\boldsymbol{v}_i$ and utilize a soft relaxation of the Bernouilli distribution [21]. The objective for each client is to maximize the variational lower bound:

$$\mathcal{L}_i(\theta) = \sum_{n=1}^{|\mathcal{D}_i|} \mathbb{E}_q \log p\left(y_i^{(n)} \middle| \boldsymbol{\phi}_i, x_i^{(n)}, W_a, W_b, \boldsymbol{r}\right) - \underbrace{\text{KL}\left(q\left(\boldsymbol{\phi}_i\right) || p\left(\boldsymbol{\phi}_i\right)\right)}_{\mathscr{R}} \tag{11}$$

$$\mathscr{R} = \sum_{\ell=1}^{L} \mathbb{E}_{q(\boldsymbol{v}_i^{\ell})}\left[\text{KL}\left(q(\boldsymbol{b}_i^{\ell}) || p(\boldsymbol{b}_i^{\ell} | \boldsymbol{v}_i^{\ell})\right)\right] + \text{KL}\left(q(\boldsymbol{v}_i^{\ell}) || p(\boldsymbol{v}_i^{\ell})\right)$$

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 (11) the first term provides label supervision and the second term ($\mathscr{R}$) regularizes the posterior with the IBP prior. The KL divergence in $\mathscr{R}$ is approximated by sampling from the posterior distribution.

C. CLIENT-SERVER COMMUNICATION

1) 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 and 2.

2) 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 factor score 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.

3) 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 [9], [11], [12], [22], [23], 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.

III. RELATED WORK

A. 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 focus on learning a single global model [1]; such an approach often suffers from model divergence, as local models may vary significantly from each other. To address this, a number of works break away from the single-global-model formulation. Several [4], [24] have cast

federated learning as a multi-task learning problem, with each client treated as a separate task. FedProx [25] 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 [26], [27]. [28] recognize performance degradation from non-i.i.d. data and propose 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 [7] and q-FFL [8] 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.

B. PRESERVING DATA SAFETY

While much progress has been made in machine learning with public datasets [14], [16], [29], in real-world settings, data are often sensitive, potentially for propriety [30], [31], security [32], or privacy [33] reasons. Protecting user data is one of the primary motivations for federated learning in the first place. Approaches include releasing artificial data [34], [35], homomorphic encryption [36], or differential privacy [37]–[39]. However, artificial data can still strongly resemble the original data, and sharing the model architecture and its parameters presents risks associated with whitebox access, leaving the data vulnerable to attacks such as membership inference [9] or model inversion [11], [12], [23].

C. BAYESIAN NONPARAMETRIC FEDERATED LEARNING

Several previous works have applied Bayesian nonparameterics to federated learning, 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 have proposed using the Beta-Bernouilli Process [40] for matching parameters, first with fully connected layers [41], but later also extended by [42] to convolutions and LSTMs [43]. In contrast, our method utilizes Bayesian nonparametrics for modeling rank-1 factors for multitask learning, instead of the aggregation stage.

D. PERSONALIZED FEDERATED LEARNING

Personalized FL models has become a recent focus. One approach is to mix the global and local model parameters during optimization [44]. However, this requires meta-features from each client, which partially violates the goal of privacy in FL. Another commonly used strategy is splitting neural networks [45], [46]: the model is divided into two parts, the feature extractor and the personalized layers. The feature extractor is aggregated and shared by the server, and both parts are trained for a personalized model. Recent work also explore meta-learning, particularly model-agnostic meta-learning (MAML) [47]. For example, Per-FedAvg [48] builds a meta-model initialization that is then updated by a gradient step for a personalized model. However, meta-optimization often requires computing second-order derivatives, which can be computationally prohibitive for FL. pFedMe [49] proposes decoupling the process of optimizing personalized models from learning the global model. pFedMe keeps the learning process of FedAvg while optimizing the personalized model in parallel, showing performance superiority over Per-FedAvg [48].

IV. EXPERIMENTS

A. EXPERIMENTAL SET-UP

1) STATISTICAL HETEROGENEITY

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.

a: UNIMODAL NON-I.I.D.

We first consider the non-i.i.d. setting introduced by [1]. 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.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.

b: MULTIMODAL NON-I.I.D.

While the *unimodal non-i.i.d.* 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. In the real world, the mode can correspond to age, gender, ethnicity, wealth, or a number of other demographic factors. The number of subpopulation group 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 [15], and animals ($N_1 = 90$) versus

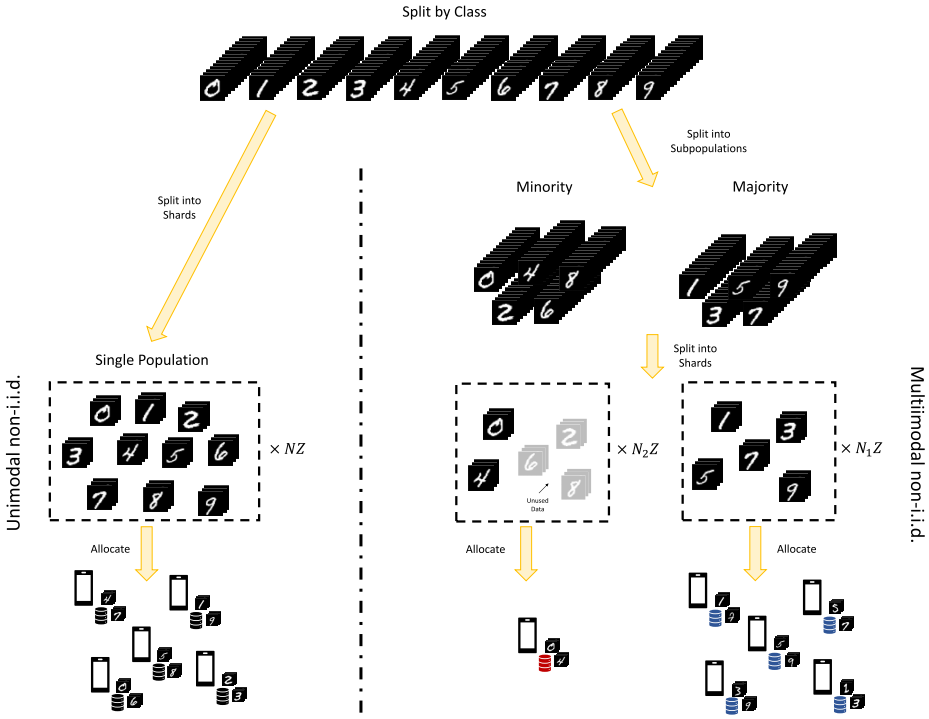


FIGURE 2. Example data allocation process to N clients for MNIST and $Z = 2$ in the unimodal i.i.d. (left) and multimodal i.i.d. (right) settings. Notice that the primary difference is the grouping of the data into two subpopulations (here referred to as "Majority" and "Minority") before sharding and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

vehicles ($N_2 = 20$) for CIFAR-10 [16], 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. An example multimodal non-i.i.d for MNIST is shown in Figure 2. Compared with unimodal non-i.i.d, 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. Note that multimodal non-i.i.d setting is not a distributed imbalanced classification setting. In multimodal non-i.i.d setting, the data amount of each class is even if aggregated while imbalanced classification setting is not. The hidden subpopulations lead to unfair model performance caused by the uniform sampling of clients.

2) MODEL ARCHITECTURE AND TRAINING SETTING

For MNIST [14] digit recognition, we use a multilayer perceptron with 1-hidden layer with 200 units using ReLU activations [50]. 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 IV-A. Stochastic gradient descent (SGD) with learning rate $\eta = 0.04$ is employed for all methods.

For FMNIST [15] 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

TABLE 1. Sub-population local test performance analysis.

	Method	MNIST	FMNIST	CIFAR-10
Major.	FedAvg	**96.63**±0.70	89.75±1.76	51.98±1.69
	FedProx	96.43±0.67	89.95±1.73	51.26±1.44
	q-FFL	94.93±0.31	88.73±0.17	42.00±0.29
	WAFFLe	95.93±0.16	88.91±2.07	**68.37**±1.01
Minor.	FedAvg	67.40±11.26	68.05±4.43	16.83±4.42
	FedProx	68.60±9.44	67.50±4.50	16.56±3.32
	q-FFL	54.20±7.37	69.40±1.48	18.14±3.05
	WAFFLe	**93.87**±0.66	**79.67**±1.52	**55.00**±6.00
Gap	FedAvg	29.23±11.79	21.70±4.21	35.15±4.12
	FedProx	27.83±10.03	22.45±4.38	32.70±6.99
	q-FFL	40.73±7.55	19.33±1.43	23.87±3.00
	WAFFLe	**2.07**±0.77	**9.25**±0.61	**13.37**±2.61
Var.	FedAvg	199±106	231±35	338±59
	FedProx	186±92	233±42	318±36
	q-FFL	355±117	212±19	220±17
	WAFFLe	**26**±6	**145**±27	**182**±27

TABLE 2. Local test performance for $Z = 2$.

	MNIST			FMNIST			CIFAR-10		
Method	Param. ↓	Unimodal ↑	Multimodal ↑	Param. ↓	Unimodal ↑	Multimodal ↑	Param. ↓	Unimodal ↑	Multimodal ↑
FedAvg	155,800	94.46±0.84	91.57±1.42	28,880	83.96±0.91	83.43±2.27	61,770	52.54±0.14	45.46±1.69
FedProx	155,800	94.44±1.15	91.53±1.05	28,800	84.19±0.99	83.59±2.30	61,770	52.36±0.11	44.95±1.17
q-FFL	155,800	91.46±1.07	88.42±1.24	28,800	83.10±0.36	85.73±0.21	61,770	43.82±0.52	38.25±1.12
WAFFLe	**120,200**	**96.23**±0.31	**95.41**±0.36	**18,155**	**87.12**±0.89	**86.09**±0.92	**42,780**	**71.30**±0.92	**66.35**±0.72

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.

For CIFAR-10 [16], 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.

For all federated learning methods, 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 $B = 10$.

For FedProx [25], the proximal parameter μ is set to 1.0. For q-FFL [8], we searched $q \in \{0.001, 0.005, 0.01, 0.1, 1, 3, 5\}$ and found $q = 0.001$ as the best setting, matching the settings of [8] on more complex data.

B. LOCAL TEST PERFORMANCE

We compare WAFFLe with FedAvg [1], the fairness-oriented q-FFL [8], and FedProx [25], which augments FedAvg with a proximal term designed for high statistical heterogeneity. We record local test performance averaged across all clients for both types of non-i.i.d. data allocation in Table 2, 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 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. Notably, 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. WAFFLe's number of communication rounds is largely the same, and by sharing rank-1 factors, each weight factor can be represented compactly, resulting in a total number of parameters that is *fewer* than the single model used by the baselines, despite using the same architecture.

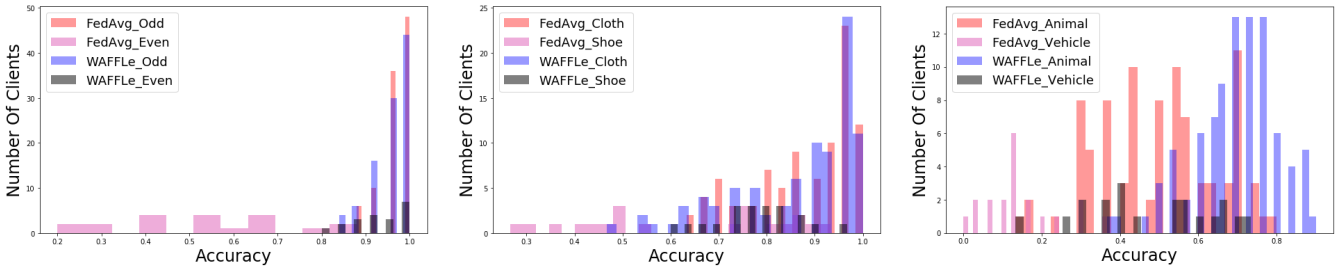
C. TRAINING EFFICIENCY COMPARISON

We plot local test accuracy against the global epoch for FedAvg, FedProx and WAFFLe on MNIST, FMNIST, and CIFAR-10 averaged over all clients for unimodal non-*i.i.d.* data in Figure 4. A similar comparison is made between FedAvg and WAFFLe for multimodal non-*i.i.d.* data in Figure 5, with the majority and minority learning curves separately shown. For both cases, the clear gap between curves shows that WAFFLe achieves better performance throughout training. Notably, WAFFLe converges at a similar rate as FedAvg with respect to the global epoch number; this is important as the number of communication rounds is often considered one of the primary bottlenecks in federated learning.

In the multimodal non-*i.i.d.* case, the difference is especially stark for the minority subpopulation, which lags significantly behind the majority when modeled with FedAvg's one-size-fits-all approach. Interesting, in addition to having lower value, the FedAvg minority's training curve is not as smooth, with large dips and spikes, especially when compared with the majority subpopulation's curve. We hypothesize that this may be due to the smaller subpopulation being more vulnerable to being unrepresented during client sampling, which may lead to catastrophic forgetting [51]. We find this to be an interesting future direction of research. In comparison, the WAFFLe minority, with its separate set of customized weight factors, has a much smoother training trajectory.

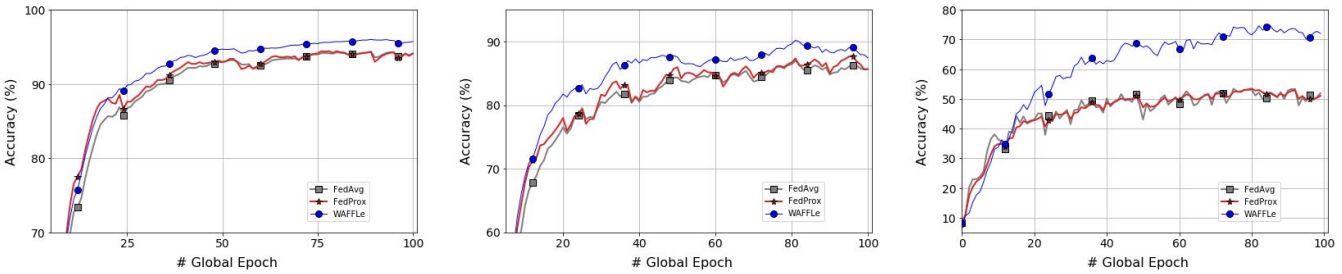
D. FAIRNESS

Average performance over all clients as in Table 2 is a commonly reported metric, but we argue that it does reveal the full story. We report subpopulation mean performance



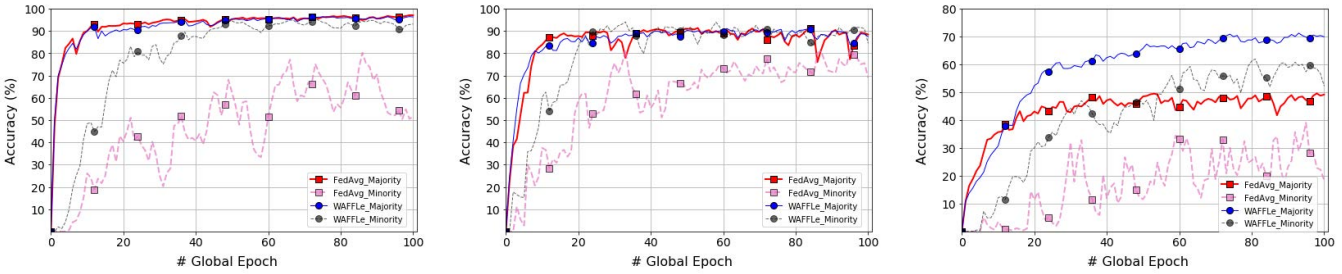
(a) (b) (c)

FIGURE 3. FedAvg and WAFFLe performance distribution across clients in the multimodal non-i.i.d. setting for (a) MNIST, (b) FMNIST and (c) CIFAR-10.



(a) (b) (c)

FIGURE 4. Local test performance for unimodal non-*i.i.d.* degree $Z = 2$. (a) MNIST; (b) FMNIST; (c) CIFAR-10.



(a) (b) (c)

FIGURE 5. Local test performance for majority and minority subpopulations for multimodal non-*i.i.d.* degree $Z = 2$. (a) MNIST; (b) FMNIST; (c) CIFAR-10.

and overall population variance in Table 1. We observe that FedAvg, which learns 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 (Major.) and minority (Minor.) values in Table 1; for example, FedAvg's performance for the ''evens'' group of clients is almost 30% lower than that of the ''odds'' group. This is gap is especially clear when visualizing the distributions of final local test performance for each client in the majority and minority groups (Figure 3). This underfitting can also be seen to exist throughout training from the ''FedAvg_Minority'' curve in Figure 5, which lags far below the ''FedAvg_Majority'' in all three datasets. On the other hand, because of WAFFLe's shared weight factor dictionary design (Equation 3), 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 [7], [8], which can be incorporated to further encourage uniform performance across clients.

E. DATA SAFETY

A primary objective of federated learning is to keep data safe. However, as mentioned in Section II-C, the predominant federated learning strategy of each client sending their entire updated model's weights still leaves the client's data vulnerable. We demonstrate this with both membership inference and model inversion attacks.

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

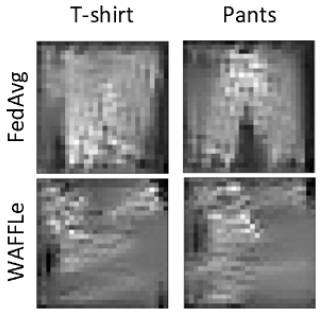


FIGURE 6. FMNIST model inversion attacks.

Membership inference attacks (MIAs) [9], [10] can be used to infer whether a given data query was used for 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 compare a LeNet [14] FedAvg [1] model with an analogous WAFFLe model, training both on 1000 CIFAR-10 samples per client. We attack both with a MIA inspired by [9], using a small ensemble of 3 "shadow" models. 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 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.

We also perform a model inversion attack [11], [23] on both FedAvg and WAFFLe. Unlike MIAs, which must start from a query data input, model inversion attacks seek to reconstruct the inputs used to train a model from the trained model itself; successful inversion attacks pose a significant risk from a data security perspective. We perform a model inversion attack on FedAvg and WAFFLe models trained on FMNIST, showing randomly selected results in Figure 6 recovered from an individual user. Importantly, reconstructions on FedAvg are significantly sharper, with the class identity far clearer than for WAFFLe, meaning FedAvg is more vulnerable to model inversion attacks.

We report two quantitative metrics [52] to evaluate model inversion attack in Table 4. *i)* Peak Signal-to-Noise Ratio (PSNR) is the ratio of an image's maximum squared pixel fluctuation over the mean squared error between the target image and the reconstructed image. The higher the PSNR, the better the quality of the reconstructed image. However, clear reconstruction images reveal the identity information of the client's data. For each class, for example T-shirt, we compute the PSNR between the reconstructed T-shirt and the average image of randomly selected T-shirt from the training data. The average PSNR of all classes of FedAvg and WAFFLe is reported. *ii)* Attack Accuracy (Attack Acc) is the accuracy of the input reconstructed image by an evaluation classifier that is trained separately. If the evaluation classifier achieves high accuracy, the reconstructed image is considered to expose identity information about the target label. We obtain an evaluation classifier with accuracy 96.67%. This evaluation classifier is used to classify the images reconstructed by FedAvg and WAFFLe. The average attack accuracy is reported. For both PSNR and attack accuracy, lower values indicate more secure method. In Table 4, WAFFLe shows superior performance over FedAvg on both PSNR and attack accuracy, proofing more secure against model-inversion attack.

TABLE 4. Data safety comparison on FMNIST.

Methods	PSNR	Attack Acc
FedAvg	12.46	60.63
WAFFLe	**10.22**	**40.78**

TABLE 5. Personalization comparison on CIFAR-10.

Methods	Unimodal	Multimodal
FedPer	67.90± 0.36	64.12 ±0.93
pFedMe	68.10± 0.2	57.16 ± 0.1
WAFFLe	**71.30** ± 0.92	**66.35** ± 0.72

F. PERSONALIZATION

We further conduct experiments to compare against two personalized FL methods FedPer [46] and pFedMe [49] based on CIFAR-10 under both unimodal and multimodal settings for $Z = 2$. The local test performance is reported in Table 5. WAFFLe outperforms FedPer by offering personalization for multiple layers while FedPer only focuses on the last layer of the neural networks. Also, WAFFLe and FedPer outperforms pFedMe by 9.19% and 6.96% under multimodal setting respectively, highlighting that the methods based on model splitting are more effective than regularization-based methods for complicated non-i.i.d. settings.

G. ABLATION STUDIES

1) 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.We also did experiments 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. The learning curve comparison when

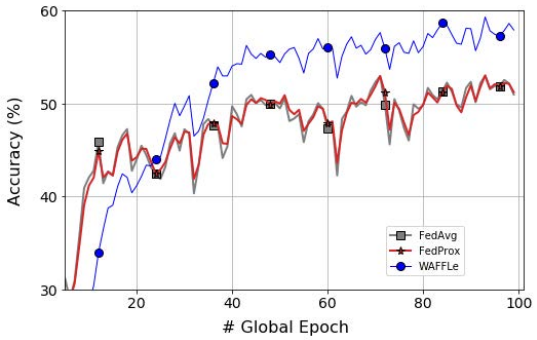


FIGURE 7. Learning efficiency comparison when $Z = 3$.

TABLE 6. Unimodal local test accuracy vs local epochs.

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

$Z = 3$ is shown in the Figure 7. Note that FedProx is based on FedAvg combining a L-2 norm on the local weights during training. The difference between FedProx and FedAvg is highly dependent on the parameter for the proximal term. We follow the setting in the previous work [25] and set the parameter as 0.2 which is not significant enough to differentiate FedProx and FedAvg in extremely non-i.i.d setting.

2) 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 6 and for multimodal non-i.i.d. in Table 7 using the same settings as in Section IV-A 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.

3) IBP 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 multimodal non-i.i.d. settings in Tables 9 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.

TABLE 7. Multimodal local test accuracy vs local epochs.

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

TABLE 8. Sparsity comparison on MNIST.

Methods	Local Test Accuracy
FedAvg	94.46
FedProx	94.44
q-FFL	91.46
WAFFLe(without L1 norm)	94.96
WAFFLe(with L1 norm, weight=0.1)	94.59
WAFFLe(with L1 norm, weight=1.0)	96.03
WAFFLe(with L1 norm, weight=2.0)	95.92
WAFFLe(with L1 norm, weight=10.0)	94.24
WAFFLE	**96.23**

TABLE 9. Multimodal local test accuracy vs α and F.

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

To empirically demonstrate the data-driven sparsity introduced by IBP prior, we also considered an alternative non-Bayesian version of our model. Specifically, we replace WAFFLe's inferred per-client weight factors with per-client weight factors optimized by standard gradient descent, and use an L1 sparsity constraint on factor usage as a replacement for the sparsity induced by the IBP. We list the test accuracy under Unimodal on MNIST in the Table 8. Note that our Bayesian formulation (WAFFLe) outperforms the non-Bayesian version while also avoiding the hyperparameter tuning of the weight of the sparsity term, which the non-Bayesian version is somewhat sensitive to.

V. CONCLUSION

We have introduced WAFFLe, a Bayesian nonparametric framework for federated learning, employing shared rank-1 weight factors. 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 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 membership inference or gradient-based model inversion attacks on the private data.

APPENDIX A GENERALIZING WEIGHT FACTORIZATION TO CONVOLUTIONAL KERNELS

While introducing WAFFLe's formulation in Section II-A, 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.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017.

[2] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," 2019, *arXiv:1908.07873*.

[3] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, Q. Le, and A. Ng, "Large scale distributed deep networks," in *Proc. Neural Inf. Process. Syst.*, 2012.

[4] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. Neural Inf. Process. Syst.*, 2017.

[5] D. Li and J. Wang, "FedMD: Heterogenous federated learning via model distillation," 2019, *arXiv:1910.03581*.

[6] D. Peterson, P. Kanani, and V. J. Marathe, "Private federated learning with domain adaptation," 2019, *arXiv:1912.06733*.

[7] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2019.

[8] T. Li, M. Sanjabi, and V. Smith, "Fair resource allocation in federated learning," in *Proc. Int. Conf. Learn. Represent.*, 2020.

[9] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017.

[10] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019.

[11] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2015.

[12] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. Neural Inf. Process. Syst.*, 2019.

[13] Z. Ghahramani and T. L. Griffiths, "Infinite latent feature models and the Indian buffet process," in *Proc. Neural Inf. Process. Syst.*, 2006.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[15] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.

[16] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[17] N. Mehta, K. J. Liang, V. K. Verma, and L. Carin, "Continual learning using a Bayesian nonparametric dictionary of weight factors," in *Proc. Artif. Intell. Statist.*, 2021.

[18] Y. W. Teh, D. Grür, and Z. Ghahramani, "Stick-breaking construction for the Indian buffet process," in *Proc. Artif. Intell. Statist.*, 2007, pp. 556–563.

[19] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2015.

[20] P. Kumaraswamy, "A generalized probability density function for double-bounded random processes," *J. Hydrol.*, vol. 46, nos. 1–2, pp. 79–88, Mar. 1980.

[21] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *Proc. Int. Conf. Learn. Represent.*, 2017.

[22] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2014.

[23] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019.

[24] L. Corinzia, A. Beuret, and J. M. Buhmann, "Variational federated multi-task learning," 2019, *arXiv:1906.06268*.

[25] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2018, *arXiv:1812.06127*.

[26] Y. Jiang, J. Konečnỳ, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," 2019, *arXiv:1909.12488*.

[27] M. Khodak, M.-F. F. Balcan, and A. S. Talwalkar, "Adaptive gradient-based meta-learning methods," in *Proc. Neural Inf. Process. Syst.*, 2019.

[28] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018, *arXiv:1806.00582*.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009.

[30] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017.

[31] C. Xu, S. Liu, Z. Yang, Y. Huang, and K.-K. Wong, "Learning rate optimization for federated learning exploiting over-the-air computation," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3742–3756, Dec. 2021.

[32] K. J. Liang, G. Heilmann, C. Gregory, S. O. Diallo, D. Carlson, G. P. Spell, J. B. Sigman, K. Roe, and L. Carin, "Automatic threat recognition of prohibited items at aviation checkpoints with X-ray imaging: A deep learning approach," *Proc. SPIE*, vol. 10632, Apr. 2018, Art. no. 1063203.

[33] D. Ribli, A. Horváth, Z. Unger, P. Pollner, and I. Csabai, "Detecting and classifying lesions in mammograms with deep learning," *Sci. Rep.*, vol. 8, no. 1, pp. 1–7, Dec. 2018.

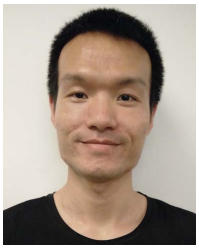
[34] A. Triastcyn and B. Faltings, "Federated generative privacy," *IEEE Intell. Syst.*, vol. 35, no. 4, pp. 50–57, Jul. 2020.

[35] J. Goetz and A. Tewari, "Federated learning via synthetic data," 2020, *arXiv:2008.04489*.

[36] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," 2017, *arXiv:1711.10677*.

[37] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. Theory Cryptogr. Conf.*, 2006, pp. 265–284.

[38] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016.
[39] L. Melis, G. Danezis, and E. De Cristofaro, "Efficient private statistics with succinct sketches," 2015, *arXiv:1508.06110*.
[40] R. Thibaux and M. I. Jordan, "Hierarchical beta processes and the Indian buffet process," in *Proc. Artif. Intell. Statist.*, 2007.
[41] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, T. N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," 2019, *arXiv:1905.12022*.
[42] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," 2020, *arXiv:2002.06440*.
[43] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[44] F. Hanzely and P. Richtárik, "Federated learning of a mixture of global and local models," 2020, *arXiv:2002.05516*.
[45] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," 2019, *arXiv:1912.00818*.
[46] Q. Yang, J. Zhang, W. Hao, G. Spell, and L. Carin, "FLOP: Federated learning on medical datasets using partial networks," 2021, *arXiv:2102.05218*.
[47] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.
[48] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning: A meta-learning approach," 2020, *arXiv:2002.07948*.
[49] C. T. Dinh, N. H. Tran, and T. D. Nguyen, "Personalized federated learning with Moreau envelopes," 2020, *arXiv:2006.08848*.
[50] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, 2010.
[51] N. Shoham, T. Avidor, A. Keren, N. Israel, D. Benditkis, L. Mor-Yosef, and I. Zeitak, "Overcoming forgetting in federated learning on non-IID data," 2019, *arXiv:1910.07796*.
[52] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song, "The secret revealer: Generative model-inversion attacks against deep neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 253–261.



WEITUO HAO is currently pursuing the Ph.D. degree in electrical and computer engineering with Duke University, Durham, NC, USA. He interned at Microsoft, in 2019, and mainly worked on vision and language representation learning. In 2020, he also interned at Samsung and worked on fairness in federated learning. He also interned at TikTok on the music recommendation systems, in 2021. His research interests include large-scale pre-training for multimodal representation learning, model compression, and federated learning. From 2020 to 2021, he was a recipient of the Samsung Fellowship.

NIKHIL MEHTA received the B.S. degree from Delhi Technological University, India. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Duke University. He interned at Google AutoLX, in 2020; Google Brain, in 2021; and Meta Platforms, in 2022. Before joining Duke, he worked as a Research Associate at IIT Kanpur. His research interests include Bayesian inference, continue learning, and novelty detection.

KEVIN J. LIANG received the B.S.E., M.S., and Ph.D. degrees from Duke University, in 2015, 2019, and 2015, respectively. He double-majored in electrical and computer engineering and biomedical engineering as an undergraduate and received his graduate degrees from the Department of Electrical and Computer Engineering, under the supervision of Lawrence Carin. He is currently a Research Scientist at Facebook AI Research (FAIR) Accel within meta platforms, where he focuses on computer vision research. While a student, he previously interned at Microsoft, in 2014; Google Cloud AI, in 2017; Infinia ML, in 2018; and Facebook, in 2019. His research interests include computer vision, continual learning, federated learning, and adversarial attacks.

PENGYU CHENG received the B.S. degree from the Department of Mathematical Sciences, Tsinghua University, in 2017, and the Ph.D. degree in electrical and computer engineering from Duke University, Durham, NC, in 2021, under the supervision of Lawrence Carin. During his Ph.D. degree, he interned at the NEC Laboratory, in 2019, and Microsoft Cloud and AI, in 2020. He is currently a Senior Researcher at the Tencent Interactive Entertainment Group. His research interests include controllable text generation and interpretable natural language understanding. He research interest includes probabilistic machine learning methods.

MOSTAFA EL-KHAMY (Senior Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Alexandria University, Egypt, the Ph.D. degree in electrical engineering from the California Institute of Technology, Caltech, USA, and the M.B.A. degree from the Edinburgh Business School, U.K. He was a founding Faculty Member of Egypt–Japan University for Science and Technology (E-JUST). Previously, he was at Qualcomm Research and Development, San Diego. He is a Senior Principal Engineer with Samsung SOC Research and Development, CA, USA. He is also an Adjunct Professor at the Faculty of Engineering, Alexandria University. His research interests include the theory and practice of artificial intelligence in multimedia and communication systems. He was a recipient of the URSI Young Scientist Award, the Caltech Atwood Fellowship, the Alexandria University Scientific Incentive Award, the Samsung Best Paper Award, and the Samsung Distinguished Inventor Award.

LAWRENCE CARIN (Fellow, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Maryland, College Park, in 1985, 1986, and 1989, respectively. In 1989, he joined the Electrical Engineering Department, Polytechnic University, Brooklyn, as an Assistant Professor, where he became an Associate Professor, in 1994. In September 1995, he joined the Electrical Engineering Department at Duke University, where he is currently a Professor. From 2011 to 2014, he was a Chairman of the Electrical and Computer Engineering Department. From 2003 to 2013, he held the William H. Younger Professorship. He was a Vice President for Research at Duke. He is a co-founder of the Signal Innovations Group, Incorporation (SIG), a small business that was acquired by BAE Systems, in 2014. He is now the Provost at the King Abdullah University of Science and Technology (KAUST). He has published over 300 peer-reviewed articles, and he is a member of the Tau Beta Pi and Eta Kappa Nu honor societies. His current research interests include machine learning and applied statistics.