Federated Learning for Commercial Image Sources

Shreyansh Jain
IIIT Delhi
New Delhi, India
shreyansh21089@iiitd.ac.in

Koteswar Rao Jerripothula
IIIT Delhi
New Delhi, India
koteswar@iiitd.ac.in

Abstract

Federated Learning is a collaborative machine learning paradigm that enables multiple clients to learn a global model without exposing their data to each other. Consequently, it provides a secure learning platform with privacy-preserving capabilities. This paper introduces a new dataset containing 23,326 images collected from eight different commercial sources and classified into 31 categories, similar to the Office-31 dataset. To the best of our knowledge, this is the first image classification dataset specifically designed for Federated Learning. We also propose two new Federated Learning algorithms, namely Fed-Cyclic and Fed-Star. In Fed-Cyclic, a client receives weights from its previous client, updates them through local training, and passes them to the next client, thus forming a cyclic topology. In Fed-Star, a client receives weights from all other clients, updates its local weights through pre-aggregation (to address statistical heterogeneity) and local training, and sends its updated local weights to all other clients, thus forming a star-like topology. Our experiments reveal that both algorithms perform better than existing baselines on our newly introduced dataset.

1. Introduction

Federated Learning (FL) is a distributed learning paradigm that can learn a global model from decentralized data without having to exchange sensitive data across the clients [24] [29].

Traditional Machine Learning requires users to upload their data to the centralized server for the learning and inference task. The end-user has no power and control over how the data is used [11]. Moreover, uploading the data to a central server incurs severe costs. Maintaining such a vast volume of data and communicating the learning parameters back to the user is costly. To overcome the privacy challenges and issue of maintaining a large amount of data in the centralized setting, the Federated Learning paradigm was proposed by Google [22], which aims to overcome these issues.

The Federated Learning framework addresses sensitive data privacy and data access issues [41]. Federated Learning models are trained via model aggregation rather than data aggregation. It requires model to be trained locally on the data owner's machine or the local edge devices, and only the model parameters are shared. Federated Learning has found successful applications in the IoT, healthcare, finance, etc [23] [31]. The traditional Federated Learning optimization methods involve local client training on the local datasets for a fixed number of epochs using an SGD optimizer. The local clients then upload the model weights to the central server, where the weights are averaged to form a global model whose parameters are shared with the local client. This method is known as FedAvg [29], which facilitates the local client to learn features from different clients while preserving privacy. However, FedAvg may have convergence issues in case the clients exhibit statistical heterogeneity, which may lead to non-convergence of the model [25] [21] [16]. Thus, a simple FedAvg algorithm may not be helpful when dealing with device-level heterogeneity.

In this work, we aim to understand the real-world scenario where different commercial image sources can collaborate in a Federated setting to perform the image classification task with privacy preservation.

Most previous research works applied the Federated Learning algorithm on a single dataset distributed among the clients in an IID or non-IID manner. This is not close to a real-world scenario where different clients may have different data distribution due to domain shift (statistical heterogeneity) among them [34] [30]. Another challenge in Federated Learning is the convergence issue when the data distribution is different among the clients, which may increase the communication cost between the clients and the central server and leads to suboptimal model performance.

Motivated by the above challenges, we propose our dataset in which each client's dataset is sampled from different commercial image sources to simulate the real-world scenario where each client exhibits a domain shift. This is

because each commercial image source has its own unique image set, causing domain shift amongst clients. The dataset is inspired by the Office-31 dataset [37]. We also propose two novel algorithms, namely Fed-Cyclic and Fed-Star. Fed-Cyclic is a simple algorithm in which, a client gets weights from the previous client, trains the model locally and passes the weights to the next client in a cyclic fashion. In this way, a global model is simply being passed from one client to another in a cyclic manner. The global server need not involve here. Even if it is required that we want to involve it to preserve anonimity, the global server does not have to perform any computation and can be used only for passing the parameters of one client to another. Fed-Star requires each client to receive weights from all the other clients during pre-aggregation in a star-like manner after the local training of each client on its train set. While pre-aggregating, every client prioritizes learning the outlier features present in different clients while retaining the common features to train a more robust model impervious to statistical heterogeneity among the client's data distribution, followed by aggregation via a global server after a fixed number of periods. Experiments show that our algorithms have better converegence and accuracy.

To summarize, our contributions are three-fold:

- We propose the an image classification dataset specifically designed for Federated Learning, which is close to a real-world scenario where each client has a unique dataset demonstrating domain shift.
- We propose Fed-Cyclic, a simple algorithm, which is communicationally efficient and attains higher accuracy than baselines.
- We propose the Fed-Star algorithm, which trains a model that prioritizes learning of generalized and outlier features to create a model personalized to each client's heterogeneous dataset distribution and attains faster convergence than the baselines with higher accuracy.

2. Related Works

Many distributed optimization algorithms have been developed to process and draw inferences from the data uploaded [47], [36], [9] [2] [1]. However, such distributed method requires uploading of data to the central server, which incurs the considerable cost of maintaining data centrally, and processing it requires a lot of power [32] [8]. Also, the privacy issue persists as the user has no control over how personal data is used and shared.

The first application of the Federating Learning algorithm is FedAvg, proposed by Mcmahan *et al*. [29]. FedAvg performs reasonably well when the data distribution is IID among the clients and shows faster convergence of the global model. The issue arises in real-world scenarios when the data follow the non-IID distribution as proposed by Zhao *et al*. [48].

Federated Learning with data heterogeneity: The vanilla FedAvg faces convergence issues when there is data heterogeneity among the clients. To tackle this challenge, different methods have been proposed. FedProx [24] adds a proximal term by calculating the square distance between the server and client with local loss to optimize the global model better. FedNova [43] proposes normalized averaging to eliminate objective inconsistency with heterogeneous Federated optimization. FedMax, as proposed by Chen *et al*. [7], aims to mitigate activation-divergence by making activation vectors of the same classes across different devices similar. FedOpt proposed by Reddi *et al*. [35] applies different optimizers to the server, like Adam, Yogi, and AdaGrad. VRL-SGD [26] incorporates variance-reduction into local SGD and attains higher accuracy while decreasing the communication cost. FedCluster [5] proposes grouping local devices into different clusters so that each cluster can implement any Federated algorithm. RingFed, proposed by Yang *et al*. [46], minimizes the communication cost by pre-aggregating the parameters among the local clients before uploading the parameter to the central server. SCAFFOLD [20] uses variance-reduction to mitigate client drift. Chen *et al*. [6] proposes FedSVRG that uses stochastic variance reduced gradient-based method to reduce the communication cost between clients and servers while maintaining accuracy. Jeong *et al*. [18] proposes Federated augmentation (FAug), which involves the local client jointly training the generative model to augment their local dataset and generate the IID dataset. In this paper, we aim to train models robust to data heterogeneity with faster convergence and lower communication costs via our proposed algorithms. One of our proposed method, Fed-Star is similar to RingFed [46]. RingFed involves simple pre-aggregation of weights between adjacent clients, whereas our method involves pre-aggregation of weights between all the local clients using the accuracy metric.

Personalized Federated Learning FedAvg also suffers from creating a generalized global model as the parameters are averaged, which gives poor representation to a client with heterogeneous data. Personalized Federated Learning involves training the global model using any Federated vanilla algorithm followed by personalizing the model for each client via locally training the model on each client [19] [28] [48] [11]. Data heterogeneity among clients is the reason for personalized Federated Learning. Data augmentation is explored to account for local data heterogeneity and involves local clients to jointly generate IID data distribution [12] [44]. Wang *et al*. [42] proposes FAVOR and selects a subset of clients at each round to mitigate the bias introduced by non-IID data. Chai *et al*. [4] pro-

poses the TiFL method to cluster the clients in different tiers and train the clients belonging to the same tier together for faster training. Sattler *et al*. [38] proposes a hierarchical clustering-based approach based on the cosine similarity of the client gradient to segment similar clients in similar clustering for training to train the local clients properly. Xie *et al*. [45] proposes Expectation Maximization to derive optimal matching between the local client and the global model for personalized training. Deng *et al*. [10] proposes APFL algorithm to find optimal client-server pair for personalized learning.There are different clustering-based approaches as explored by different authors [3] [14] [17] [13]. Tan *et al*. [40] provides deeper analysis of personalized Federated framework. In our work, we aim to personalize the model at the global level by proposing an algorithm that better captures the outlier features of the client while retaining the generalized features.

Federated Image classification datasets Most Federated Learning algorithms are simulated on datasets belonging to a single domain with an artificial partition among the clients or use existing public datasets. The dataset distribution may differ for different clients in the real-world scenario as the clients exhibit domain shift. The first work proposing the real-world image dataset [27] contains more than 900 images belonging to 7 different object categories captured via street cameras and annotated with detailed boxes. The image dataset has applications in object detection. In our work, we propose the first real-world image dataset for the image classification task to better understand the performance of the Federated algorithm in a real-world setting.

3. Proposed Methods

3.1. Objective

In Federated Learning (FL), different clients (say K clients) collaborate to learn a global model without having to share their data. Let the weights of such a model be w, and let the loss value of the model for sample (x_i, y_i) be $\mathcal{L}(x_i, y_i; w)$. The objective now is to find optimal w such that the following objective is achieved:

$$\min \frac{1}{|D|} \sum_{i=1}^{|D|} \mathcal{L}(x_i, y_i; w) \tag{1}$$

where D denotes the union of all the data owned by different clients, as shown below:

$$D = \bigcup_{k=1}^{K} D_k \tag{2}$$

where D_k denotes the data owned by the k^{th} client. Given this, we can rewrite our objective function as follows:

$$\min \frac{1}{|D|} \sum_{k=1}^{K} \sum_{i=1}^{|D_k|} \mathcal{L}(x_i, y_i; w) \tag{3}$$

If we represent the average local loss $\mathbf{L}_k$ of k^{th} client using

$$\mathbf{L}_k = \frac{1}{|D_k|} \sum_{i=1}^{|D_k|} \mathcal{L}(x_i, y_i; w), \tag{4}$$

we can reformulate the objective as follows:

$$\min \sum_{k=1}^{K} \frac{|D_k|}{|D|} \mathbf{L}_k \tag{5}$$

which suggests that our objective is to minimize the weighted sum of local losses incurred by our clients, and the weights are proportional to the number of data samples clients have with them. This objective function is similar to same as [29]. We discussed it to make our paper self-contained.

This formulation motivated FedAvg [29] to aggregate the local model weights in the weighted averaging manner to obtain w as shown below:

$$w = \sum_{k=1}^{K} \frac{|D_k|}{|D|} w_k \tag{6}$$

This aggregation happens iteratively, where, in a given iteration, the central server sends the global model to the local clients, where local models are updated and then sent back to the global server for aggregation, as shown in Figure 1. This happens until the global model converges.

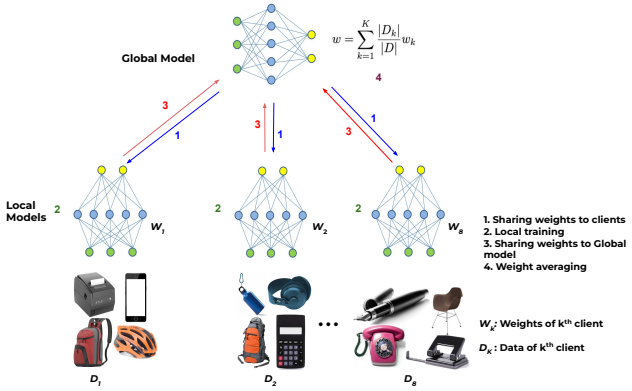

Figure 1. FedAvg algorithm [29]

However, it can take several communication rounds for the FedAvg algorithm to converge, especially when there is statistical heterogeneity in clients' datasets. As a result,

the accuracy drops too [15]. Also, FedAvg creates a generalized model by averaging the parameters from the local clients, forcing the local model with statistical heterogeneity to learn a generalized representation that may differ from its data distribution, leading to poorly trained local clients. In that case, local clients do not find the global model satisfactory.

Considering these limitations of FedAvg, we propose Fed-Cyclic and Fed-Star algorithms, which cater to the statistical heterogeneity of data across the clients and ensure the satisfactory local performance of global models despite that.

3.2. Fed-Cyclic

We propose the Fed-Cyclic algorithm to overcome the challenges faced by the FedAvg algorithm, which suffers from a communication bottleneck due to a large number of edge devices uploading the parameters to the central server, which causes congestion in the network. The model visualization is shown in Figure 2

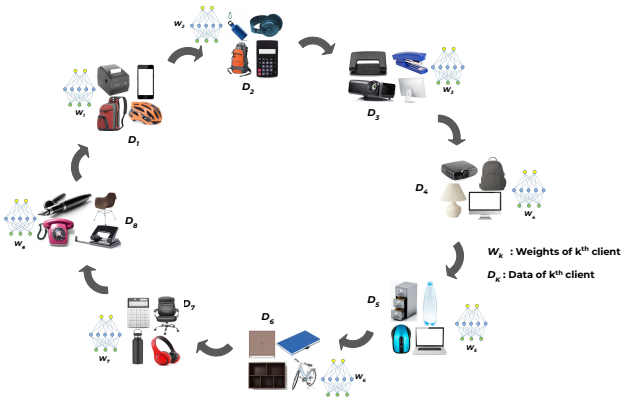

Figure 2. Proposed model using Fed-Cyclic showing passing of weights cyclically by the clients after local training.

In the Fed-Cyclic algorithm, we use a global model to initialize the weight of one of the clients in the network, followed by training the client's local model for E local epochs. The optimizer used is SGD at the local client. The updated weights are then used to initialize the weight of the next client in the network, as shown in equation (7), and the process continues until all the clients are trained cyclically in this manner, constituting one training round. In our Fed-Cyclic algorithm, the clients can either directly pass the weights to the next client or involve the global server to do so to preserve the anonymity of the last client. After the end of each round, the global weights w get updated.

It is a communication-efficient algorithm since the clients can directly pass the weights to the next client without involving the global server. Even if the global server is involved in passing the weights from one client to another, no processing is done, and only parameters from a single client are passed at a time. We explain it in Algorithm 1 in greater detail. The most important step is the following:

Algorithm 1 Proposed Fed-Cyclic Algorithm

Input Initial weights w_{init}, number of global rounds R, number of local epochs E, learning rate η, K clients indexed by k (with local data D_k and local weights w_k) and local minibatch size b.

Output Global weights w^R (after R rounds)

Algorithm:

Initialize $w^0 \leftarrow w_{init}$ // Global weights initialized
$w_1^0 \leftarrow w^0$
for r=*0 to* $R-1$ **do**
  for k=*1 to* $K-1$ **do**
    $w_k^{r+1} \leftarrow ClientUpdate(w_k^r, k)$ using SGD
    $w_{k+1}^r \leftarrow w_k^{r+1}$
  $w_K^{r+1} \leftarrow ClientUpdate(w_K^r, K)$ using SGD
  $w_1^{r+1} \leftarrow w_K^{r+1}$
  $w^{r+1} \leftarrow w_K^{r+1}$
function $ClientUpdate(w, k)$
  $B \leftarrow$ (split D_k into batches of size b)
  for e=1 to E **do**
    for $d \in B$ **do** $w \leftarrow w$ - $\eta \nabla$g(w;d)
  return w

$$w_{k+1}^r \leftarrow w_k^{r+1} \tag{7}$$

where we use k^{th} client to initialize $(k+1)^{th}$ client.

The algorithm is robust to statistical heterogeneity as every client gets an opportunity to train the global model on the local data. Moreover, we can take the view that the global model is being periodically trained on different portions of the dataset (D), as if they are mini-batches (D_k). Hence, this algorithm is somewhat analogous to a typical deep learning approach from the point of view of the global model. As a result, convergence also gets ensured, unlike FedAvg, where we expect convergence for simple aggregation of weights.

3.3. Fed-Star

Although Fed-Cyclic can converge faster than FedAvg, it is a very simple algorithm, similar to FedAvg. Also, it lacks aggregation of any kind. Here, we propose the Fed-Star algorithm, where we address these limitations.

In Fed-Star, for some time, the local models are trained locally for some epochs in a parallel manner. Once the given epochs are complete, we perform pre-aggregation of weights locally at each client by sharing their models with each other. Each client gets models from every other client, and they are evaluated on the local training set of the given client. The accuracy obtained helps us determine how much

Algorithm 2 Proposed Fed-Star Algorithm

Input: Initial weights w_{init}, number of global rounds R, number of local epochs E, learning rate η, K clients indexed by k (with local data D_k and local weights w_k), local minibatch size b, number of period P, and weight matrix M of dim $K * K$.

Output: Global weights w^R (after R rounds)

Algorithm:

Initialize $w^0 \leftarrow w_{init}$ // Global weights initialized

for r=0 to $R-1$ **do**

  $w_k^{r,0} \leftarrow w^r, \forall k \in \{1, \cdots, K\}$

  for $p \in \{, \cdots, P-1\}$ **do**

    for $k \in \{1, \cdots, K\}$ **parallely do**

      $w_k^{r,p+1} \leftarrow ClientUpdate\ (w_k^{r,p}, k)$

      for $j \in \{1, \cdots, K\}$ **do**

        transfer $w_j^{r,p+1}$ to k^{th} client

        $M(k,j) = 1 - Acc(w_j^{r,p+1}, D_k)/100$

      $w_k^{r,p+1} = \frac{\sum_{j=1}^{K} M(k,j) * w_j^{r,p+1}}{\sum_{j=1}^{K} M(k,j)}$

  $w^{r+1} = \frac{1}{K} \sum_{k=1}^{K} \frac{|D_k|}{|D|} w_k^{r,P}$

function $ClientUpdate(w, k)$

  $B \leftarrow$ (split D_k into batches of size b)

  for e=1 to E **do**

    for $d \in B$ **do** $w \leftarrow w$ - $\eta \nabla$g(w;d)

  return w

function $Acc(w, D)$

  return Accuracy of w on train set of D

weightage should be given to the models of each client during pre-aggregation. These pre-aggregated weights are now used to reinitialize the local model for training. These steps are iteratively carried out, and these iterations are called periods. This interaction for model sharing is analogous to star network topology, where every client interacts with every other client in a network. After a certain number of periods P, the local weights are aggregated on the central servers, so a round has P periods in it, where local models are being shared with each other, they are getting pre-aggregated at each client for initialization to train the model in the next period.

In any period p of round r, a weightage matrix M gets developed, which is computed as follows:

$$M(k,j) = 1 - Acc(w_j^{r,p+1}, D_k)/100 \tag{8}$$

where $Acc(w_j^{r,p+1}, D_k)$ denotes the training accuracy of $w_j^{r,p+1}$ on training set of dataset D_k. It denotes the weightage value for the model coming from j^{th} model while pre-aggregating at k^{th} client.

Figure 3. Proposed model using Fed-Star demonstrating pre-aggregation of the parameters in star topology manner among local clients and is given by equation (9). This is followed by the transferring of weight to the global model where aggregation of weights is performed.

Note here that we give more weightage to the client significantly different from the reference client during pre-aggregation because we want each client to learn the outlier features from the clients while retaining the generalized features during pre-aggregation.

The pre-aggregated weights for k^{th} client are depicted as follows:

$$w_k^{r,p+1} = \frac{\sum_{j=1}^{K} M(k,j) * w_j^{r,p+1}}{\sum_{j=1}^{K} M(k,j)} \tag{9}$$

where we normalize the weightages with their sum while performing the pre-aggregation of weights.

Thus, each local model tends to learn more from the other client that is significantly different. The Fed-Star algorithm is explained further in Algorithm 2 and can be visualized through Figure 3.

This algorithm is communication intensive since each client has to interact with every other client. Still, the total communication overhead is reduced significantly as the pre-aggregation step among clients decreases the reliance on the global server for convergence. Our algorithm attains faster convergence than FedAvg with lesser communication overhead with the global server and higher accuracy. Fed-Star retains outlier features well and helps create a global model that is also personalized to the local clients.

4. Proposed Dataset

We propose a dataset containing 23,326 images which we collected from 8 different image-hosting websites. Each of the 8 sources represents 8 different clients in our Federated learning setting. The average number of images

Figure 4. Sample images from our dataset.

Classes	Mean	Std. dev.	Total Img.
back_pack	108.50	37.98	868
bike	108.62	31.73	869
bike_helmet	94.25	26.54	754
book_shelf	77.13	25.93	617
bottle	89.38	11.61	715
calculator	111.50	31.43	892
desk_chair	125.38	27.61	1003
desk_lamp	106.87	24.87	855
desktop_computer	99.00	19.13	792
file_cabinet	65.38	20.13	523
headphone	105.63	22.36	845
keyboard	76.38	35.14	611
laptop	111.38	25.39	891
letter_tray	26.38	13.47	211
mobile_phone	84.88	15.06	679
monitor	112.38	37.35	899
mouse	116.87	25.97	935
mug	117.38	25.19	939
notebook	98.00	23.78	784
pen	108.50	25.20	868
phone	113.63	21.80	909
printer	108.25	22.64	866
projector	72.63	33.86	581
puncher	66.88	32.37	535
ring_binder	65.63	31.26	525
ruler	90.13	19.50	721
scissors	127.13	51.77	1017
speaker	67.63	18.6	541
stapler	105.25	21.75	842
tape_dispenser	82.38	37.9	659
trashcan	103.25	33.36	826

Table 1. The mean, standard deviation and the total number of images in the classes across different commercial sources.

for each source is approximately 2916, divided across 31 categories. This dataset is inspired from the Office-31 dataset [37], which contains common objects in the office settings like keyboard, printer, monitor, laptop, and so forth. Our dataset includes the same categories of images as were in the Office-31 dataset. We took extra care to ensure that only relevant and high-quality images from each source were taken. We removed poor quality, duplicate or irrelevant images by manually curating the dataset. The statistics showing how images are distributed within the classes and across the sources are summarized in Tables 1 and 2. The sample images are shown in Figure 4.

Source	Mean	Std. dev.	Total Img.
123rf	94.16	26.96	2897
Adobe Stock	101.16	27.25	3104
Alamy	102.80	40.65	3155
CanStockPhotos	95.06	33.24	2915
Depositphotos	101.41	38.59	3112
Getty Images	63.06	21.99	1923
iStock	90.10	33.01	2761
Shutterstock	112.61	36.34	3459

Table 2. The mean, standard deviation of images distribution across the classes of the dataset together with total images in each source.

5. Experiments

5.1. Implementation Detail

In this section, we describe the experiments we performed to evaluate our Federated image classification algorithms. We leverage the pretrained VGG-19 [39] network available from PyTorch pretrained model library [33] for initialization purpose. We freeze its convolutional layers and replace the rest of the network with three new fully connected layers (of size 1024, 256 and 31) and a softmax layer. In the first two fully connected layers, we use ReLU activation and a dropout rate of 0.5. We have used SGD optimizer with a batch size of 64. We use the 80:20 train-test split of the data at any client. The evaluation metric used is classification accuracy, but we have also evaluated using Macro F1 score and weighted F1 score. The default learning rate is 3e-4 ($3x10^{-4}$). The different iteration parameters used are given in Table 3.

5.2. Results

We evaluate all four algorithms (FedAvg [29], RingFed [46], Fed-Cyclic, Fed-Star) on our dataset. For RingFed, we kept γ=0.8, as we obtained the best accuracy for RingFed using this value, as shown in Table 4.

We provide results of both global evaluation and local evaluation. While local test sets are used for local evaluation, their union is used for global evaluation. As we can see in Table 5, where we provide the global evaluation results,

Method	Local Epochs (E)	Period (P)	Global Rounds (R)
FedAvg	3	-	250
RingFed	3	2	50
Fed-Cyclic	3	-	150
Fed-Star	3	2	50

Table 3. Iteration Parameters

γ	Accuracy
0.2	89.22%
0.5	88.96%
0.8	**89.65%**
1.0	89.39%

Table 4. γ vs accuracy for RingFed

Method	Accuracy	Weighted F1	Macro F1
FedAvg [29]	89.11%	88.98%	88.47%
RingFed [46]	89.65%	89.53%	89.14%
Fed-Cyclic (ours)	91.15%	90.89%	90.33%
Fed-Star (ours)	91.72%	91.17%	90.58%

Table 5. Experimental results show that both the Fed-Star and Fed-Cyclic attains higher accuracy than FedAvg and F1-scores. Here, red denotes the best value and blue denotes the second best value.

our two proposed methods, Fed-Cyclic and Fed-Star, perform better than FedAvg and RingFed. Fed-Star performs the best among the four, with an accuracy of 91.72%. Moreover, our methods converge very fast. Fed-Star requires 50 global rounds on our dataset, as mentioned in Table 3. Although RingFed also requires the same number of global rounds, its accuracy is lower than Fed-Star.

In Table 6, where we provide the results of the local evaluation, we compare our methods with competing Federated Learning methods and the baseline of the respective local model (using $E = 250$). As can be seen, our Fed-Star algorithm gets the best results on all the clients. Our Fed-Cyclic algorithm comes second in 6 out of 8 cases, losing to the local model on Depositphotos and iStock. As far as FedAvg and RingFed are concerned, they lose to 5 and 4 local clients, respectively. This suggests that FedAvg and RingFed do not help much from a personalization point of view.

5.3. Learning Rate (η) Experiments

We also performed our experiments while varying the learning rate. For FedAvg, we have observed that accuracy steadily increases with the decrease in the learning rate, and maximum accuracy is obtained for the learning rate of 3e-3 with the value of 91.43%. For RingFed, we observed that

Dataset	Model	Test Accuracy
123rf	Local Model	85.51%
	FedAvg	83.79%
	RingFed	84.47%
	Fed-Cyclic (Ours)	86.38%
	Fed-Star (Ours)	88.90%
Adobe Stock	Local Model	92.43%
	FedAvg	91.46%
	RingFed	92.07%
	Fed-Cyclic (Ours)	94.52%
	Fed-Star (Ours)	94.96%
Alamy	Local Model	86.84%
	FedAvg	87.32%
	RingFed	87.98%
	Fed-Cyclic (Ours)	88.90%
	Fed-Star (Ours)	90.14%
CanStockPhotos	Local Model	89.53%
	FedAvg	89.20%
	RingFed	89.56%
	Fed-Cyclic (Ours)	90.40%
	Fed-Star (Ours)	91.68%
Depositphotos	Local Model	98.07%
	FedAvg	96.95%
	RingFed	97.76%
	Fed-Cyclic (Ours)	97.91%
	Fed-Star (Ours)	98.33%
Getty Images	Local Model	89.35%
	FedAvg	90.13%
	RingFed	91.06%
	Fed-Cyclic (Ours)	92.72%
	Fed-Star (Ours)	93.87%
iStock	Local Model	85.71%
	FedAvg	83.72%
	RingFed	84.68%
	Fed-Cyclic (Ours)	84.89%
	Fed-Star (Ours)	86.47%
Shutterstock	Local Model	89.45%
	FedAvg	89.60%
	RingFed	90.16%
	Fed-Cyclic (Ours)	91.56%
	Fed-Star (Ours)	92.11%

Table 6. Fed-Star outperforms all the local models trained using traditional ML method and baselines and Fed-Cyclic on different sources.

the value of accuracy increased with an increase in learning rate from 1e-3 to 7e-3 (91.81% to 92.17%), followed by dropping in accuracy for the learning rate of 3e-4 to 89.65%. For the Fed-Cyclic algorithm, maximum accuracy is obtained for the learning rate of 3e-3 with an accuracy value of 92.52% and minimum accuracy of 91.15% for the learning rate of 3e-4. Fed-Star attains maximum accuracy of 92.77% for a learning rate of 1e-3. The accuracy drops with an increase in the learning rate, falling to the value of 91.72% for the learning rate of 3e-4. The detailed results are captured in Figure 5 and Table 7.

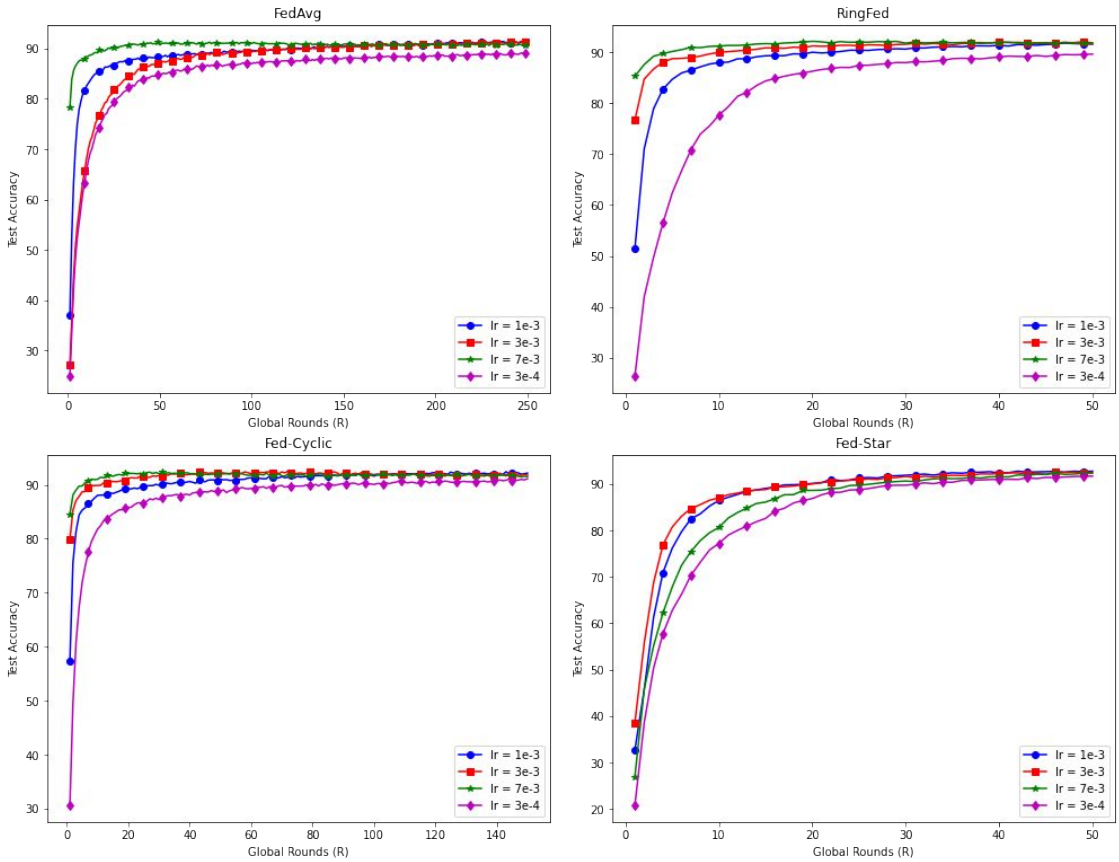

Figure 5. The graph shows how the accuracy of FedAvg, RingFed, Fed-Cyclic and Fed-Star changes with different learning rates (lr).

Learning Rate (η)	FedAvg [29]	RingFed [46]	Fed-Cyclic (Ours)	Fed-Star (Ours)
1e-3	91.39%	91.81%	92.42%	92.77%
3e-3	91.43%	92.09%	92.52%	92.68%
7e-3	91.33%	92.17%	92.35%	92.53%
3e-4	89.11%	89.65%	91.15%	91.72%

Table 7. The table shows the accuracy value after convergence attained by FedAvg, RingFed, Fed-Cyclic and Fed-Star for different values of learning rates.

Conclusion

We proposed two Federated Learning algorithms, Fed-Cyclic and Fed-Star, along with a new federated image classification dataset collected from 8 commercial image sources, making the setup much closer to a real-world scenario than other image classification setups where an existing dataset itself is artificially divided. Our algorithms have better convergence and better accuracy than FedAvg and RingFed algorithms. Also, they perform much better from the personalization point of view, making them very relevant for meaningful collaboration amongst clients having statistical heterogeneity (domain shift).

References

[1] Mohammad Amiri-Zarandi, Rozita A Dara, and Evan Fraser. A survey of machine learning-based solutions to protect privacy in the internet of things. *Computers & Security*, 96:101921, 2020.

[2] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[3] Christopher Briggs, Zhong Fan, and Peter Andras. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2020.

[4] Zheng Chai, Ahsan Ali, Syed Zawad, Stacey Truex, Ali Anwar, Nathalie Baracaldo, Yi Zhou, Heiko Ludwig, Feng Yan, and Yue Cheng. Tifl: A tier-based federated learning system. In *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, pages 125–136, 2020.

[5] Cheng Chen, Ziyi Chen, Yi Zhou, and Bhavya Kailkhura. Fedcluster: Boosting the convergence of federated learning via cluster-cycling. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 5017–5026. IEEE, 2020.

[6] Dawei Chen, Choong Seon Hong, Yiyong Zha, Yunfei Zhang, Xin Liu, and Zhu Han. Fedsvrg based communication efficient scheme for federated learning in mec networks. *IEEE Transactions on Vehicular Technology*, 70(7):7300–7304, 2021.

[7] Wei Chen, Kartikeya Bhardwaj, and Radu Marculescu. Fedmax: Mitigating activation divergence for accurate and communication-efficient federated learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 348–363. Springer, 2020.

[8] Emiliano De Cristofaro. An overview of privacy in machine learning. *arXiv preprint arXiv:2005.08679*, 2020.

[9] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13(1), 2012.

[10] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

[11] Georgios Drainakis, Konstantinos V Katsaros, Panagiotis Pantazopoulos, Vasilis Sourlas, and Angelos Amditis. Federated vs. centralized machine learning under privacy-elastic users: A comparative analysis. In *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*, pages 1–8. IEEE, 2020.

[12] Moming Duan, Duo Liu, Xianzhang Chen, Renping Liu, Yujuan Tan, and Liang Liang. Self-balancing federated learning with global imbalanced data in mobile systems. *IEEE Transactions on Parallel and Distributed Systems*, 32(1):59–71, 2020.

[13] Moming Duan, Duo Liu, Xinyuan Ji, Renping Liu, Liang Liang, Xianzhang Chen, and Yujuan Tan. Fedgroup: Efficient federated learning via decomposed similarity-based clustering. In *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pages 228–237. IEEE, 2021.

[14] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 33:19586–19597, 2020.

[15] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[16] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[17] Li Huang, Andrew L Shea, Huining Qian, Aditya Masurkar, Hao Deng, and Dianbo Liu. Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records. *Journal of biomedical informatics*, 99:103291, 2019.

[18] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

[19] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[20] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[21] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for on-device federated learning. 2019.

[22] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[23] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020.

[24] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[25] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

[26] Xianfeng Liang, Shuheng Shen, Jingchang Liu, Zhen Pan, Enhong Chen, and Yifei Cheng. Variance reduced local sgd with lower communication complexity. *arXiv preprint arXiv:1912.12844*, 2019.

[27] Jiahuan Luo, Xueyang Wu, Yun Luo, Anbu Huang, Yunfeng Huang, Yang Liu, and Qiang Yang. Real-world image datasets for federated learning. *arXiv preprint arXiv:1910.11089*, 2019.

[28] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

[29] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[30] Jose G Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. *Pattern recognition*, 45(1):521–530, 2012.

[31] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 23(3):1622–1658, 2021.

[32] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. Towards the science of security and privacy in machine learning. *arXiv preprint arXiv:1611.03814*, 2016.

[33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[34] Joaquin Quinonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. Mit Press, 2008.

[35] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

[36] Peter Richtárik and Martin Takáč. Distributed coordinate descent method for learning with big data. *The Journal of Machine Learning Research*, 17(1):2657–2681, 2016.

[37] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010.

[38] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*, 32(8):3710–3722, 2020.

[39] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[40] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[41] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM workshop on artificial intelligence and security*, pages 1–11, 2019.

[42] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 1698–1707. IEEE, 2020.

[43] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.

[44] Qiong Wu, Xu Chen, Zhi Zhou, and Junshan Zhang. Fedhome: Cloud-edge based personalized federated learning for in-home health monitoring. *IEEE Transactions on Mobile Computing*, 2020.

[45] Ming Xie, Guodong Long, Tao Shen, Tianyi Zhou, Xianzhi Wang, Jing Jiang, and Chengqi Zhang. Multi-center federated learning. *arXiv preprint arXiv:2005.01026*, 2020.

[46] Guang Yang, Ke Mu, Chunhe Song, Zhijia Yang, and Tierui Gong. Ringfed: Reducing communication costs in federated learning on non-iid data. *arXiv preprint arXiv:2107.08873*, 2021.

[47] Sixin Zhang, Anna E Choromanska, and Yann LeCun. Deep learning with elastic averaging sgd. *Advances in neural information processing systems*, 28, 2015.

[48] 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.