Federated Learning for Commercial Image Sources

by
Shreyansh Jain

Under the supervision of
Dr. Koteswar rao Jerripothula

Indraprastha Institute of Information Technology - Delhi
May, 2023

Federated Learning for Commercial Image Sources

by
Shreyansh Jain

Submitted in partial fulfilment of the requirements for the degree of Master of Technology

to

Indraprastha Institute of Information Technology - Delhi
May, 2023

Certificate

This is to certify that the thesis titled *"Federated Learning for Commercial Image Sources"* being submitted by **Shreyansh Jain** to the Indraprastha Institute of Information Technology Delhi, for the award of the Master of Technology, is an original research work carried out by him under my supervision. In my opinion, the the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

May, 2023

Dr Koteswar Rao Jerripothula

Department of Computer Science
Indraprastha Institute of Information Technology Delhi
New Delhi 110 020

Acknowledgements

I would like to thank Dr. Koteswar Rao Jerripothula for his continuous support and guidance. Without his motivation and guidance, the work would not have been possible. Since day one, he has been very much supportive towards me. My driving force was his impeccable guidance and constant push to work hard and do quality work. He always motivated me to do better. He was always available as a mentor whenever I felt stuck. I am grateful to work under him and learn much professionally and personally.

I sincerely thank my parents for their unconditional love and support. They believed in me and helped me immensely in innumerable ways. They constantly motivated me and always backed me up in my endeavours. I am here only because of their belief, unconditional love and affection. I am truly indebted to them for everything.

I would also like to thank my friend Lavanya, whom I reached to whenever I felt down or stressed. She was as much invested in the success of work as I was. I am genuinely indebted for everything she has done for me to make my work possible. She unconditionally gave so much of her time and energy to help me directly or indirectly whenever I felt helpless and stuck with my work. This work couldn't have been completed without her. I would also like to thank Abhay and Prakhar, who were always available to listen to my rants and motivate me. I could only make it to this college because of their motivation and belief. I thank everyone who helped me out in all the ways possible.

Publications

This is a list of published/submitted works as part of my M.tech thesis.

1. **Shreyansh Jain**, and Koteswar Rao Jerripothula, "Federated Learning for Commercial Image Sources." In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (**WACV**), 2023, Waikoloa, Hawaii, United States of America

Abstract

With the growing awareness regarding user privacy and the demand for transparency, traditional distributed machine learning algorithms are becoming taboo that uses the user's data without any restriction or privacy. The corporation uses this paradigm to harvest user data to gain more insights into user interaction with their application to provide a better experience. However, the end-user has no control over the private data used by these companies. This necessitates the adoption of a decentralized paradigm in which the data is safe with the user. Federated Learning (FL) is a distributed learning paradigm that can learn a global model from decentralized data without exchanging sensitive data across the users. Through our work, we intend to study the application of Federated Learning in the domain of Computer Vision for image classification. Through our thesis, 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 distributions due to domain shifts. To address this, we propose our own dataset derived from 8 different commercial sources to understand the application of Federated Learning in real-world scenarios and understand how different commercial sources can collaborate in a federated setting. Also, 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. For this part, we specifically worked on the Domain shift issue. To tackle this, we worked on proposing two novel methods, namely Fed-Cyclic and Fed-Star. In the final part of our work, we worked on the problem of class-label imbalance among different clients and explored different techniques to mitigate the issue and create robust models that can effectively learn from non-IID data.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 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 [6] [2].

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 [7]. 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 [8], which aims to overcome these issues.

The Federated Learning framework addresses sensitive data privacy and data access issues [9]. 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 [10] [11]. 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 [2], 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 [12] [13] [3]. Thus, a simple FedAvg algorithm may not be helpful when dealing with device-level heterogeneity. The working is shown in figure 1.2.

In this work, we aim to understand the real-world scenario where different com-

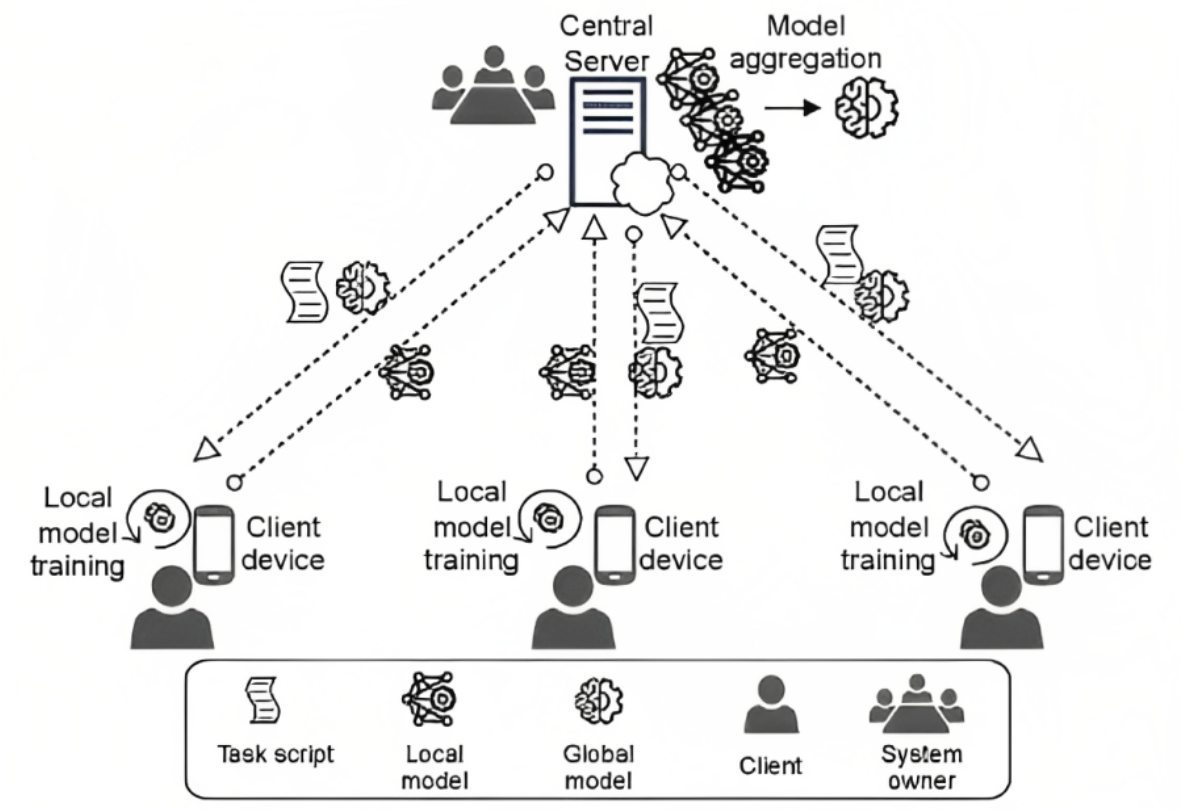

Figure 1.1: High level architecture of Federated Learning [1]. In this method, clients train the model locally and only share the weight with the central server, where the weights are aggregated and shared back with the clients without any data sharing or privacy violation. This enables the local clients to learn from one another without sharing the data.

mercial image sources can collaborate in a Federated setting to perform the image classification task with privacy preservation.

1.1 Background

With the advent of technology, more and more users are connecting to the internet and mass-media consumption has increased. People are using different applications such as Youtube, Netflix, amazon etc., depending on their needs in large numbers. However, the companies behind these applications also want to provide a personalized experience

to the user for monetary benefit. To do so, they analyze the user-shared data with unrestricted access to understand the interaction pattern. However, this may lead to gross privacy violations since users cannot control how these organizations use or share their data. With increased awareness, users want more control over how their data is used. This necessitates the adoption of a decentralized paradigm in which the data is safe with the user. On the other, the paradigm should facilitate the organization to leverage the user data without any privacy violation to provide an improved experience. Federated Learning (FL) is a distributed learning paradigm that can learn a global model from decentralized data without exchanging sensitive data across the users. The global model can help organizations better understand the usage pattern while the data is safe with the end user.

1.2 Motivation

We are the first to explore a possible collaboration amongst commercial image sources to carry out image classification using Federated Learning for image classification .

Figure 1.2: Collaboration among different commercial sources via Federated Learning for image classification.

We aim to study the Federated Learning methodology in the computer vision domain, where we want to understand the application and practicality of the paradigm in real-world scenarios. 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 [14] [15]. 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 contains 23,326 images collected from eight different commercial sources and classified into 31 categories, similar to the Office-31 dataset [5]. 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 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 anonymity, the global server does not have to perform any computation. It can be used only to pass one client's parameters 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. This is followed by using Fed-Cyclic on the pre-existing datasets to understand how well our algorithm performs on the different scenarios of non-IID distribution schemes.

1.3 Experiments

We have performed the image classification task using our own dataset, inspired by the Office-31 dataset. We collected close to 24,000 images distributed among 31 different classes and performed the task of image classification method using baselines and our own proposed methods. We also extend the experimentation to the Office-31 dataset to further show that our models generalize well when handling the task of domain

shift among the clients' data distribution. We further performed experimentation to demonstrate personalized model training using the Fed-Star algorithm. Finally, to conclude our research work, we concluded our experimentation by dividing the data in a non-IID manner using Dirichlet distribution among the clients and proposing Fed-Cyclic combined with label averaging that can mitigate the issue of imbalanced class label distribution among the clients on the CIFAR-10 dataset beating the baselines.

1.4 Contributions

To summarize, our contributions are five-fold:

- We are the first to explore a possible collaboration among commercial image sources to carry out image classification using Federated Learning for image classification.
- We propose the 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 communicationally efficient simple algorithm that 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.
- We propose Fed-Cyclic algorithm combined with label averaging that can handle non-IID distribution of data in terms of imbalanced class label distribution among the clients and performs better than the baseline when tested on CIFAR-10 dataset.

Chapter 2

Related Works

Many distributed optimization algorithms have been developed to process and draw inferences from the data uploaded [16], [17], [18] [19] [20]. 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 [21] [22]. 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. [2]. 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. [23].

2.1 Federated Learning with data heterogeneity

The vanilla FedAvg faces convergence issues when data heterogeneity exists among the clients. To tackle this challenge, different methods have been proposed. FedProx [6] adds a proximal term by calculating the square distance between the server and client with local loss to optimize the global model better. FedNova [24] proposes normalized averaging to eliminate objective inconsistency with heterogeneous Federated optimization. This method considers the heterogeneous work done by the different clients and optimizes the weight updation of the global model, considering the variational work done by each client. FedMax, as proposed by Chen et al. [25], aims to mitigate activation divergence by making activation vectors of the same classes across different devices. To achieve this, FedMax uses a prior based on the principle of maximum entropy. This prior assumes little knowledge about the activation prior and intends to make them

similar across different classes for better learning. FedOpt proposed by Reddi et al. [26] applies different optimizers to the server, like Adam, Yogi, and AdaGrad and shows how different optimizers affect the convergence of the global model on heterogeneous data in the non-convex setting. VRL-SGD [27] incorporates variance-reduction into local SGD, decreasing the communication cost among different clients and increasing accuracy. This method also incorporates parallelization in which multiple devices can be trained parallelly to accelerate the training of the models. FedCluster [28] proposes grouping local devices into different clusters so that each cluster can implement any Federated algorithm. In this method, models are trained cyclically within each cluster. Therefore, each learning round of FedCluster consists of multiple cycles of meta-update that boost the overall convergence. RingFed, proposed by Yang et al. [4], minimizes the communication cost by pre-aggregating the parameters among the local clients without involving the global server. After performing multiple cycles of pre-aggregation, the model updates the parameters to the global client for global aggregation. Pre-aggregation among local clients reduces the communication overhead with global clients. SCAFFOLD [29] uses variance-reduction to mitigate client drift among the local updates and attains faster convergence and higher accuracy with lower communication cost in comparison to FedAvg. Chen et al. [30] proposes FedSVRG that uses stochastic variance reduced gradient-based method to reduce the communication cost between clients and servers while maintaining accuracy. This paper deals with reducing the cost of uploading the parameters to the central server and reduces the problem to a finite-sum optimization problem. This is followed by using the FedSVRG method that decreases the number of iterations between the participants and server from the system perspective while maintaing the accuracy. Jeong et al. [31] proposes Federated augmentation (FAug), which involves the local client jointly training the generative model to augment their local dataset and generate the IID dataset. This paper aims 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 [4]. 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.

2.2 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 train-

ing the model on each client [32] [33] [23] [7]. 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 jointly generating IID data distribution [34] [35]. Wang et al. [36] proposes FAVOR, an experience-driven control framework that intelligently chooses the client devices to participate in each round of federated learning to counterbalance the bias introduced by non-IID data and to speed up convergence. This is followed by a proposal of a mechanism based on deep Q-learning that learns to select a subset of devices in each communication round to maximize a reward that encourages the increase of validation accuracy and penalizes the use of more communication rounds. Chai et al. [37] proposes the TiFL method to cluster the clients in different tiers to avoid the issue of staggler clients. In this method, devices are grouped into different tiers based on resource and data quantity heterogeneity. TiFL uses an adaptive tier selection strategy to adjust the tiering on the fly based on the observed training performance and accuracy over time to tame further the heterogeneity brought on by non-IID(Independent and Identical Distribution) data and resources. TiFL accomplishes substantially faster training performance while maintaining (and occasionally improving) overall test accuracy. Sattler et al. [38] proposes Clustered Federated Learning (CFL), a Federated Multi-Task Learning (FMTL) framework that groups the client population into clusters with concurrently trainable data distributions by utilising geometric characteristics of the FL loss surface. CFL applies to general non-convex objectives and doesn't require any changes to the FL communication mechanism. Clustering is only done once Federated Learning has reached a stationary point; therefore, CFL can be thought of as a post-processing technique that, by enabling clients to create more specialised models, will always perform better than standard FL. Xie et al. [39] proposes a multi-centre aggregation mechanism to cluster clients using their models' parameters. It learns multiple global models from data as the cluster centres and simultaneously derives the optimal matching between users and centres. We then formulate it as an optimization problem that can be efficiently solved by a stochastic expectation maximization (EM) algorithm Deng et al. [40] proposes the APFL algorithm to create a mixture of local and global models optimally to attain the personalization. There are different clustering-based approaches as explored by different authors [41] shows that it may not be optimal to train a single global model when faced with the non-IID distribution. The paper proposes a hierarchical clustering step to separate clusters of clients by the similarity of their local updates to the global joint model. Once separated into different clusters, clusters are trained separately and parallelly, allowing more clients to attain higher accuracy. [42] captures the settings where different groups of users have their own objectives but by aggregating their data with others in the same cluster (same learning task), they can leverage the

strength in numbers to perform more efficient federated learning. [43] applies Federated Learning in a clinical context. The paper applies the community-based federated learning method to group the data into clinically meaningful communities creating a global model for each community. CBFL outperformed the baseline federated machine learning. [44] proposes a clustered federated learning (CFL) framework FedGroup, in which the training of clients is grouped based on the similarities between the clients' optimization directions for high training performance, followed by the construction of a new data-driven distance measure to improve the efficiency of the client clustering procedure. The algorithm also implements a newcomer device cold start mechanism based on the auxiliary global model for framework scalability and practicality. FedGroup can achieve improvements by dividing joint optimization into groups of sub-optimization and can be combined with FL optimizer FedProx attaining higher accuracy than the SOTA CFLs. Tan et al. [45] provides a deeper analysis of a 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.

2.3 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 [46] 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.

2.4 Non-IID distribution techniques

[2] proposes the pathological distribution of the data in which each client has data from only two classes. [3] proposes the synthetic non-identical client data distribution in which every client training example are drawn independently with class labels following a categorical distribution over N classes parameterized by a vector q. Here, To synthesize a population of non-identical clients, we draw q $\sim$ Dir(p) from a Dirichlet distribution, where p characterizes a prior class distribution over N classes, and $\alpha >$

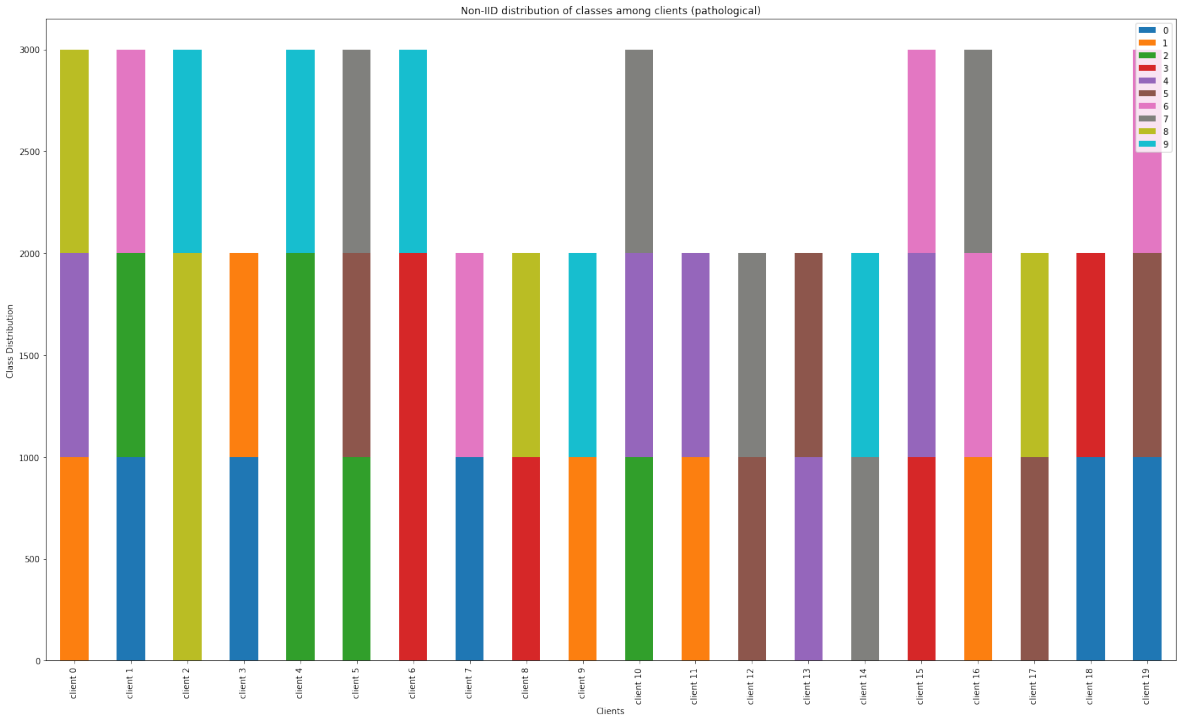

Figure 2.1: Depiction of Pathological data distribution as implemented in [2].

0 is a concentration parameter controlling the identicalness among clients. Figure 2.1 and Figure 2.2 show the two types of data distribution. We have covered the use case of domain shift among different clients. We also studied different non-IID distribution techniques covering the use case of imbalanced class distribution among the clients to understand how our proposed model works. We found out that our model Fed-Cyclic combined with label averaging performed better than SOTA techniques.

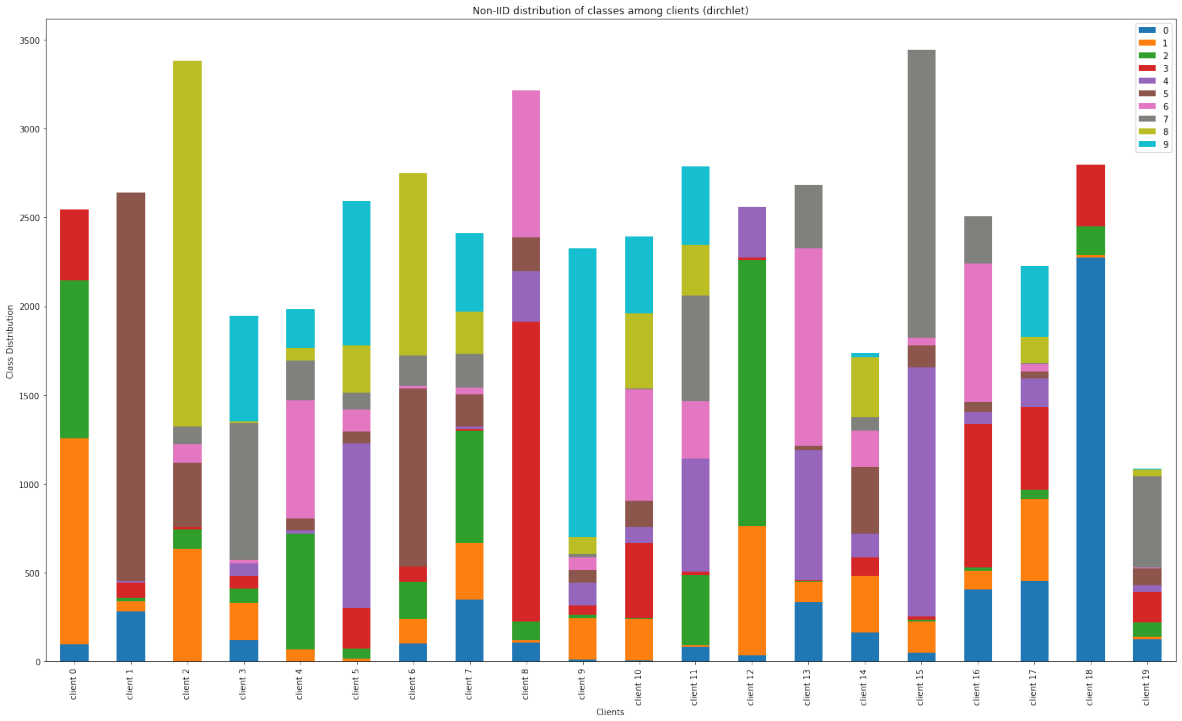

Figure 2.2: Depiction of Dirchlet data distribution as implemented in [3].

Chapter 3

Proposed Methodology

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:

$$\min \frac{1}{|D|} \sum_{i=1}^{|D|} \mathcal{L}(x_i, y_i; w) \tag{3.1}$$

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

$$D = \bigcup_{k=1}^{K} D_k \tag{3.2}$$

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

$$\min \frac{1}{|D|} \sum_{k=1}^{K} \sum_{i=1}^{|D_k|} \mathcal{L}(x_i, y_i; w) \tag{3.3}$$

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

$$\mathbf{L}_k = \frac{1}{|D_k|} \sum_{i=1}^{|D_k|} \mathcal{L}(x_i, y_i; w), \tag{3.4}$$

we can reformulate the objective as follows:

$$\min \sum_{k=1}^{K} \frac{|D_k|}{|D|} \mathbf{L}_k \tag{3.5}$$

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 the same as [2]. We discussed it to make our work self-contained.

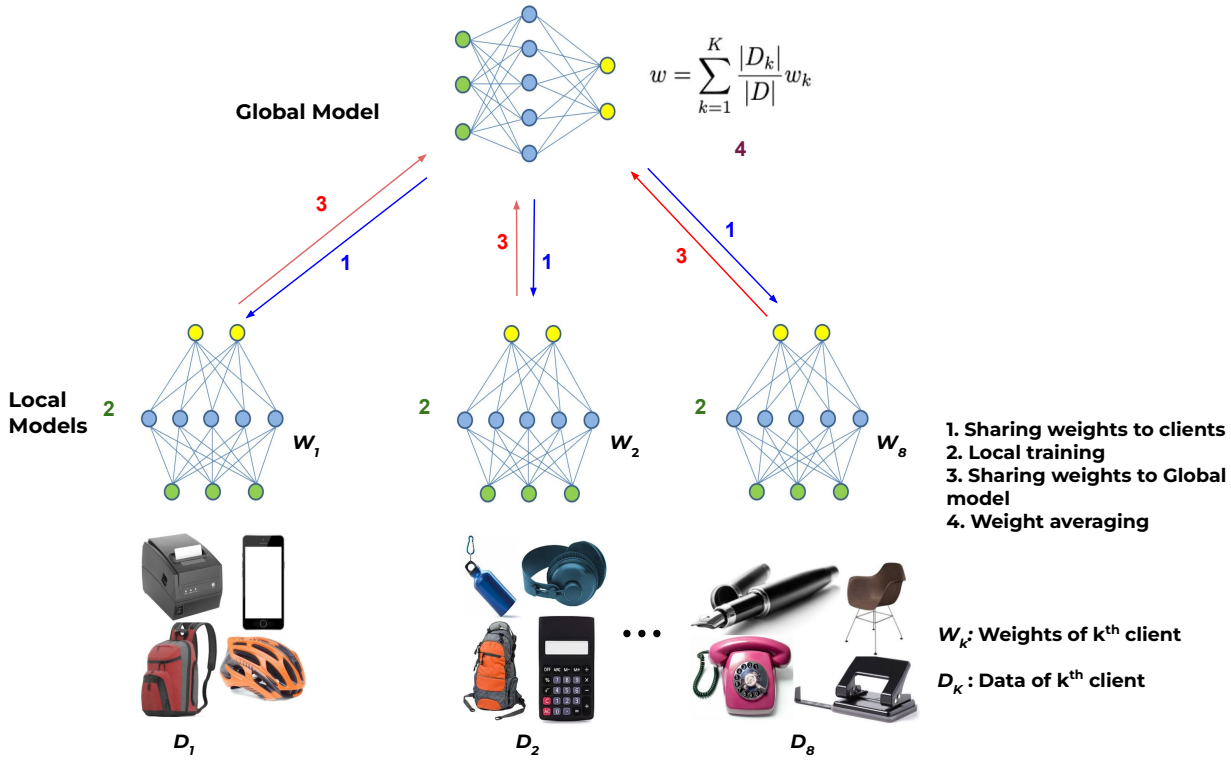

Figure 3.1: FedAvg algorithm [2]

This formulation motivated FedAvg [2] to aggregate the local model weights in the weighted averaging manner to obtain w as shown below:

$$w = \sum_{k=1}^{K} \frac{|D_k|}{|D|} w_k \tag{3.6}$$

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 3.1. This happens until the global model converges.

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 [47]. 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 3.2

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 (3.7), 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:

$$w_{k+1}^{r} \leftarrow w_{k}^{r+1} \tag{3.7}$$

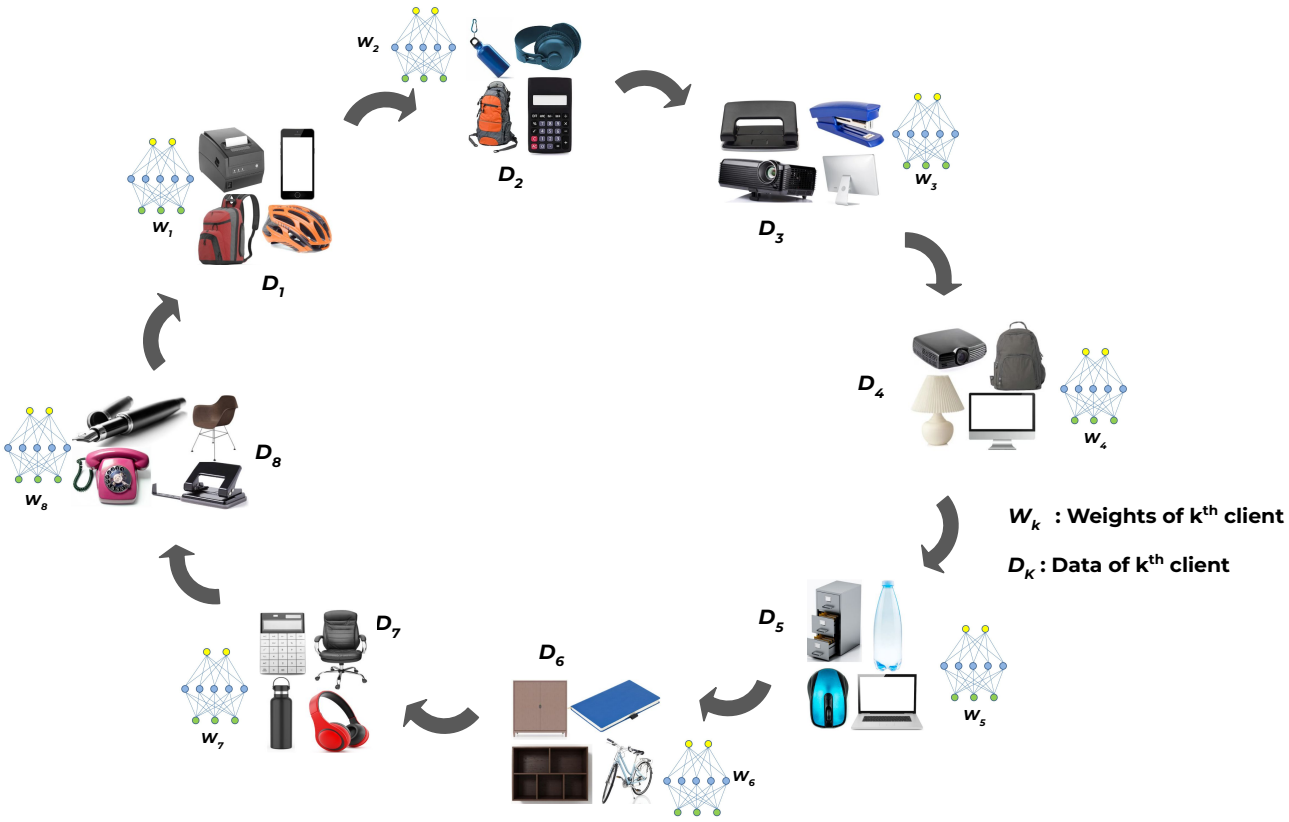

Figure 3.2: Proposed model using Fed-Cyclic showing passing of weights cyclically by the clients after local training.

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.

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

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 client's local training set. The accuracy obtained helps us determine how much 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.

3.3.1 Pre-aggregation at local level for Personalization

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

$$M(k,j) = 1 - Acc(w_j^{r,p+1}, D_k)/100 \tag{3.8}$$

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.

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:

$$w_k^{r,p+1} = \frac{\sum_{j=1}^{K} M(k,j) * w_j^{r,p+1}}{\sum_{j=1}^{K} M(k,j)} \tag{3.9}$$

Figure 3.3: Proposed model using Fed-Star demonstrating pre-aggregation of the parameters in star topology manner among local clients and is given by equation (3.9). This is followed by the transferring of weight to the global model where aggregation of weights is performed.

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 3.4.

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.

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

3.3.2 Comparison with RingFed

RingFed [4] also involves pre-aggregation of the client's weight to attain faster convergence. Still, their pre-aggregation technique involves the adjacent clients exchanging weights with each other controlled by a hyperparameter γ which decides how much weightage should be given to the weights of the adjacent client. The same is show in the In Fed-Star, we pre-aggregate the weights for a completely different reason. We perform pre-aggregation to learn the outlier features and generalized features done locally by the clients. Through our work, we intend to attain faster convergence with better accuracy by meaningfully addressing the statistical heterogeneity of data distribution among clients. Although we also use a pre-aggregation step, our objective and formulation are completely different.

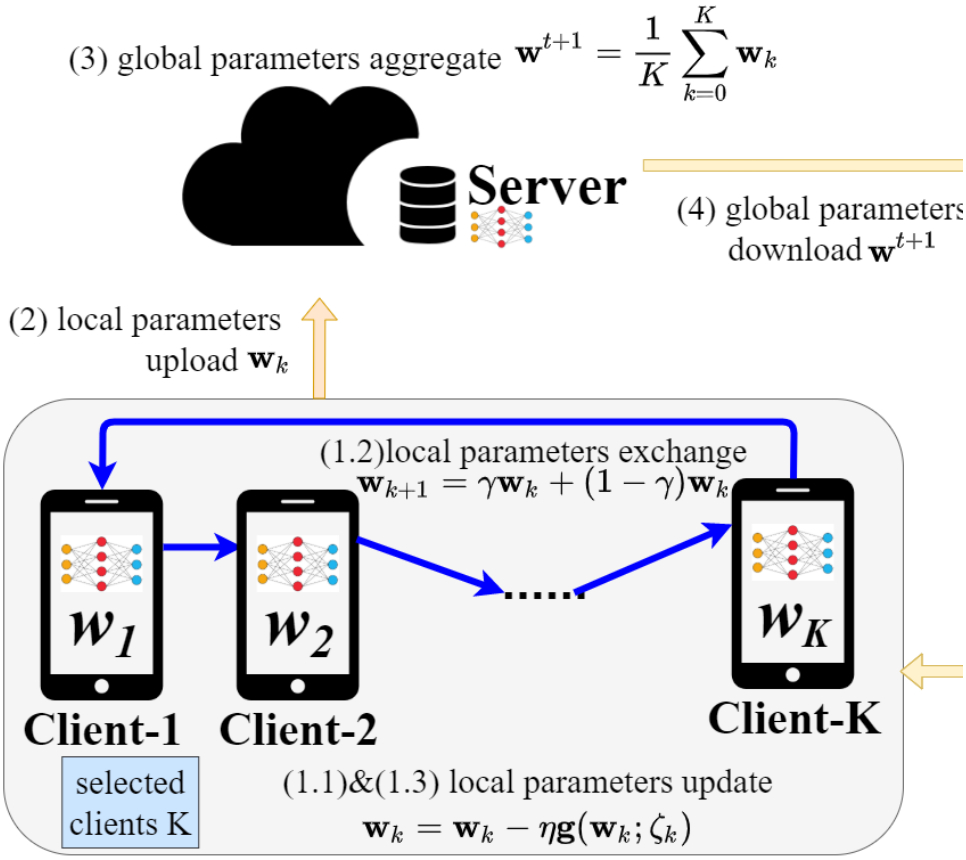

Figure 3.4: The images shows the pre-aggregation at the local level in the RingFed [4].

3.4 Fed-Cyclic with label-averaging

The above algorithms perform very well in case the client shows a shift in the domain. However, to handle the case of non-IID distribution due to class label imbalance among the client.

In this method, all the clients will not share the data but share the number of images they have for each class with the global server. Once the global server receives class-distribution statistics for each client, it calculates each class's global label distribution average and passes it back to each client. This step is performed once.

This method is a slight modification of the Fed-Cyclic method. Here, a client receives the updated weights from the previous client. Before going for the local training, the client will first compare how many samples of each label it has with the average label distribution across the network. If the samples of a class are lesser, it will resample it making it equal to the global label distribution average for that class. Else it will proceed directly. In this manner, even minority classes will get a better representation in the training phase. This method ensures that the trained local model will focus on classifying each class better and that the global model formed after the global aggregation of local weights is also robust to classify each class more efficiently in case of non-IID data distribution. The working of the algorithm is depicted in figure 3.5, which shows how each client shares their class label distribution with the global server, which calculates the global average distribution vector and shares it back with the client. Each client uses this global distribution vector during training to randomly resample minority classes with distribution less than the global average class distribution in conjugation with Fed-Cyclic to mitigate the issue of non-IID data. This is shown in figure 3.6. The algorithm is explained in 3

Algorithm 3 Proposed Fed-Cyclic Algorithm with Label averaging

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), Global model G,number of classes N,global vector $gvec[n]$ of dimension n, class vector $C[n]$ of dimension n for each client k and local minibatch size b.

Output Global weights w^R (after R rounds)

Algorithm:

Initialize $w^0 \leftarrow w_{init}$ // Global weights initialized
$w_1^0 \leftarrow w^0$
gvec $\leftarrow GlobalAverageVectorCalculation(n, k)$
for r=*0 to* $R-1$ **do**
  for k=*1 to* $K-1$ **do**
    for n=*1 to* N **do**
      if $C[n]_k < gvec[n]$ **then**
        randomly sample x images as X_n
        $D_k^{'} \leftarrow D_k \cup X_n$
      $w_k^{r+1} \leftarrow ClientUpdate(w_k^r, k)$ 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
function $GlobalAverageVectorCalculation(n, k)$
  for k=*1 to* K **do**
    for n=*1 to* N **do**
      $gvec[n] \leftarrow gvec[n] + C[n]_k$
  $gvec \leftarrow gvec \div K$
  return $gvec$

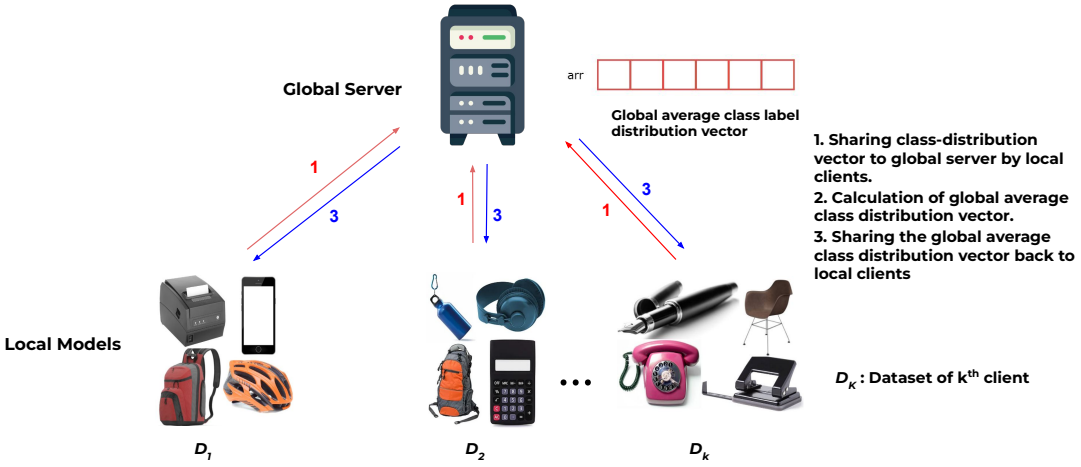

Figure 3.5: The image shows the calculation global average class label distribution calculation to be used by each class in conjugation with the Fed-Cyclic to overcome the issue of non-IID data.

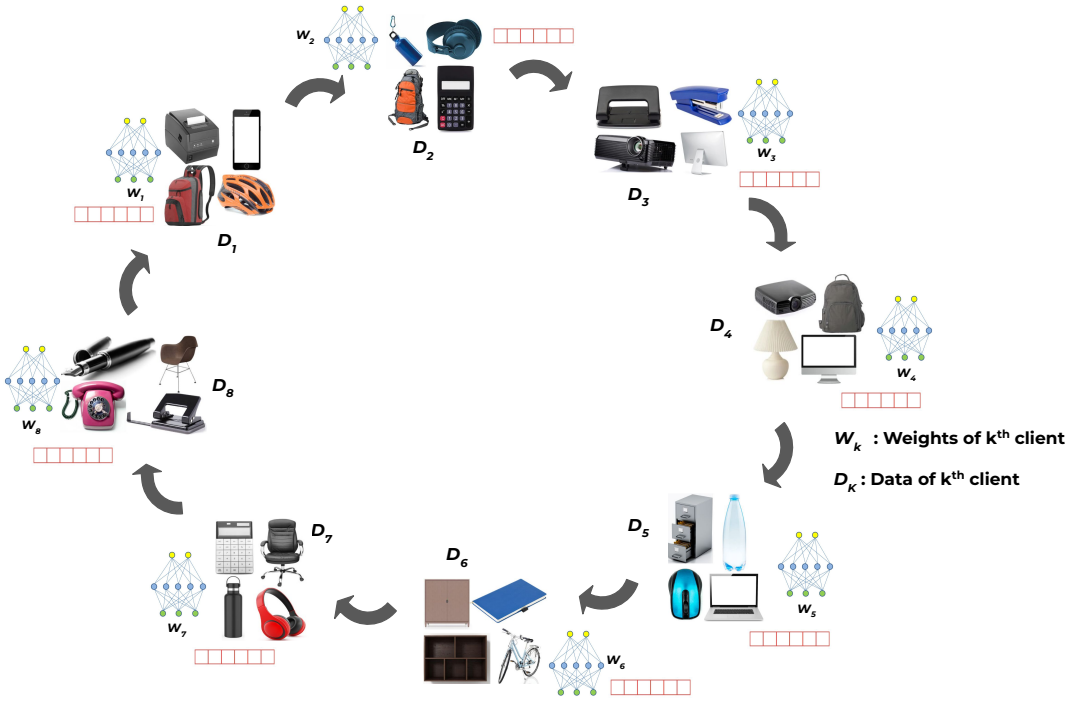

Figure 3.6: The image working of Fed-Cylic with label averaging with the help of global average distribution vector.

Chapter 4

Datasets

4.1 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. Each source's average number of images is approximately 2916, divided across 31 categories. This dataset is inspired from the Office-31 dataset [5], which contains common objects in the office settings like keyboard, printer, monitor, laptop, and so forth. Our dataset includes the same categories of images in the Office-31 dataset. We took extra care to ensure that only relevant and high-quality images from each source were taken. We manually curated the dataset by removing poor quality, duplicate or irrelevant images. The statistics showing how images are distributed within the classes and across the sources are summarized in Tables 4.1 and 4.2. The sample images are shown in Figure 4.1.

4.2 Other datasets

Office-31

We have also performed the same experiments we performed with our dataset on the original Office-31 dataset. The Office dataset contains 31 object categories in three domains: Amazon, DSLR and Webcam. The categories are objects in office settings like keyboards, mouse, phone etc. The Amazon domain contains on average 90 images per class and 2817 images in total. The merchant captured these images against a clean white background in uniform resolution. The DSLR domain contains 498 low-noise, high-resolution images with dimensions of 4288×2848. There are 5 objects per

Figure 4.1: 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 4.1: The mean, standard deviation and the total number of images in the classes across different commercial sources.

category. Each object was captured from different viewpoints on average 3 times. For Webcam, the 795 images of low resolution (640×480) exhibit significant noise and colour as well as white balance artefacts. The sample images from the dataset are shown in figure 4.2.

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

Figure 4.2: Sample images from Office-31 dataset [5].

CIFAR-10

The CIFAR-10 dataset consists of 60,000 images of dimension 32x32. The images are coloured images distributed in 10 classes, with 6000 images per class. The number of training images is 50,000 and the number of testing images is 10,000. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training

batches contain the remaining images in random order, but some training batches may have more images from one class than another. Between them, the training batches contain exactly 5000 images from each class. The sample images from the dataset are shown in figure 4.3.

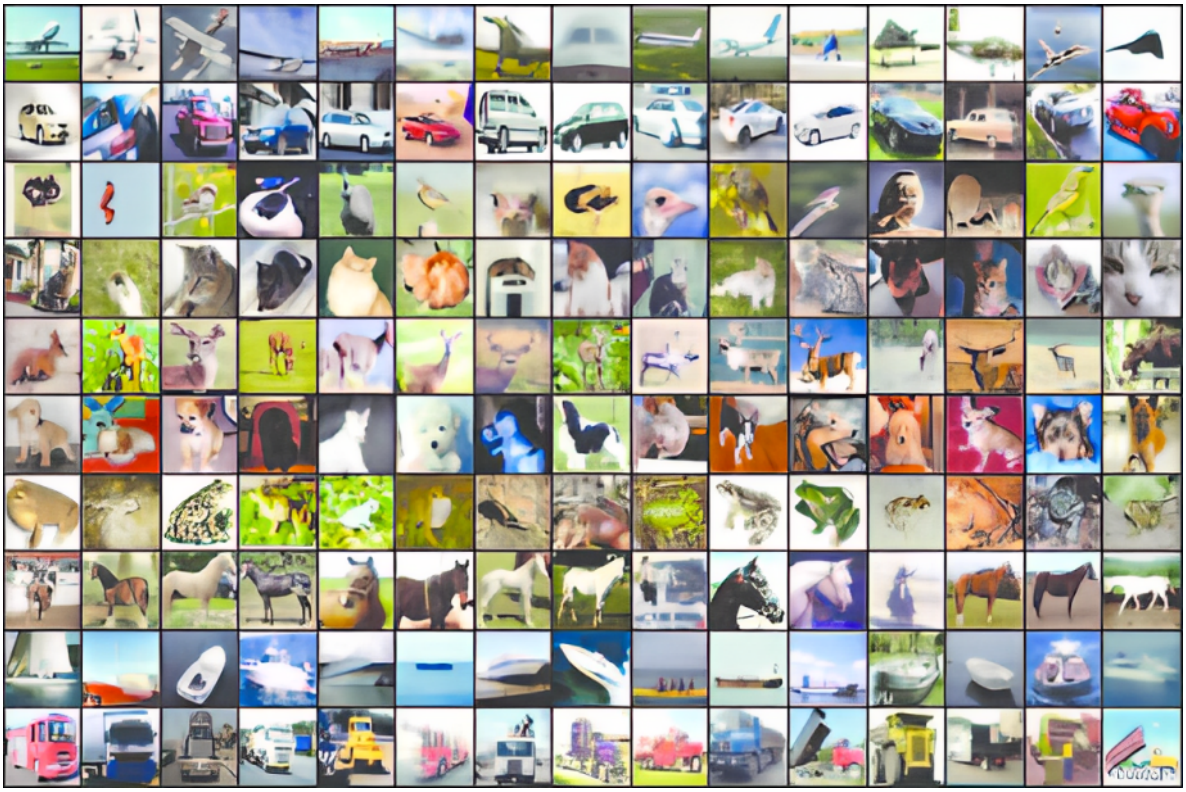

Figure 4.3: Sample images from CIFAR-10 dataset.

Chapter 5

Experimentation and Results

5.1 Implementation details

Domain Shift

In this section, we describe the experiments we performed to evaluate our Federated image classification algorithms on the task of Domain shift. We leverage the pretrained VGG-19 [48] network available from PyTorch pretrained model library [49] 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 images were resized to 224 x 224. 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 ($3x10^{-4}$). These experiments were performed on our proposed dataset and the Office-31 dataset. The different iteration parameters used are given in Table 5.1.

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 5.1: Iteration Parameters for Office-31 dataset and our proposed dataset

Non-IID (Class label imbalance)

In this section, we describe the experiments we performed to evaluate our Federated image classification algorithms using CIFAR-10 dataset divided among the clients in non-IID fashion. We leverage the VGG-19 [48] network available from PyTorch [49] for initialization purpose. We use the convolutional layer as it is and replace the rest of the network with three new fully connected layers (of size 512, 512 and 10) and a softmax layer. In the first two fully connected layers, we use ReLU activation. We have used SGD optimizer with a batch size of 32 for the train set and batch size 1000 for the test set. 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 0.01. The different iteration parameters used are given in Table 5.2.

Method	Local Epochs (E)	Global Rounds (R)
FedAvg	5	150
Fed-Cyclic	5	150
Fed-Cyclic w/ L.A.	5	150

Table 5.2: Iteration Parameters on CIFAR-10 dataset

5.2 Results

5.2.1 Domain-shift task on our proposed dataset

We evaluate all four algorithms (FedAvg [2], RingFed [4], Fed-Cyclic, Fed-Star) on our proposed dataset. For RingFed, we kept γ=0.8, as we obtained the best accuracy for RingFed using this value, as shown in Table 5.3.

We provide results of both global evaluation and local evaluation on our proposed dataset. While local test sets are used for local evaluation, their union is used for global evaluation. As we can see in Table 5.9, where we provide the global evaluation results, 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 5.1. Although RingFed also requires the same number of global rounds, its accuracy is lower than Fed-Star.

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

Table 5.3: γ vs accuracy for RingFed

Method	Accuracy	Weighted F1	Macro F1
FedAvg [2]	89.11%	88.98%	88.47%
RingFed [4]	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 5.4: Experimental results show that both the Fed-Star and Fed-Cyclic attain higher accuracy than FedAvg and F1-scores on our proposed dataset. Here, red denotes the best value and blue denotes the second best value.

Method	Accuracy	Weighted F1	Macro F1
FedAvg [2]	83.71%	82.48%	88.24%
RingFed [4]	84.85%	84.29%	83.94%
Fed-Cyclic (ours)	85.42%	85.16%	85.23%
Fed-Star (ours)	90.28%	89.57%	89.38%

Table 5.5: Experimental results show that both the Fed-Star and Fed-Cyclic attain higher accuracy than FedAvg and F1-scores on the office-31 dataset as well which shows our model generalizes well in domain-shift scenario. Here, red denotes the best value and blue denotes the second best value.

To show that our methods generalize well for the task of domain shift, we also evaluated our proposed methods on the Office-31 dataset. We sampled the classes repeatedly to keep the average number of classes in Webcam domain. Our evaluation shows that The two baselines FedAvg and RingFed attain accuracies of 83.71% and 84.85%. Regarding our proposed algorithms Fed-Cyclic and Fed-Star are concerned, they perform

Clients	Accuracy
3	87.58%
4	88.08%
6	88.72%
8	89.11%

Table 5.6: No. of clients vs accuracy for FedAvg

better with accuracy of 85.42% and 90.28%, respectively. The hyperparameters remain the same as the ones mentioned in table 5.1 with a learning rate of 3e-4. The results are tabulated in table 5.5.

As part of benchmarking our proposed dataset, We started with the FedAvg algorithm and tested how the model performance varies with changing the number of clients. For FedAvg, the accuracy steadily decreases as we sample fewer clients out of a pool of 8 clients for the given no. of epochs, with accuracy dropping to 87.58% when 3 clients are sampled in each training round. The accuracy value for different values of clients for FedAvg is summarized in Table 5.6. The same is shown in figure 5.1 We

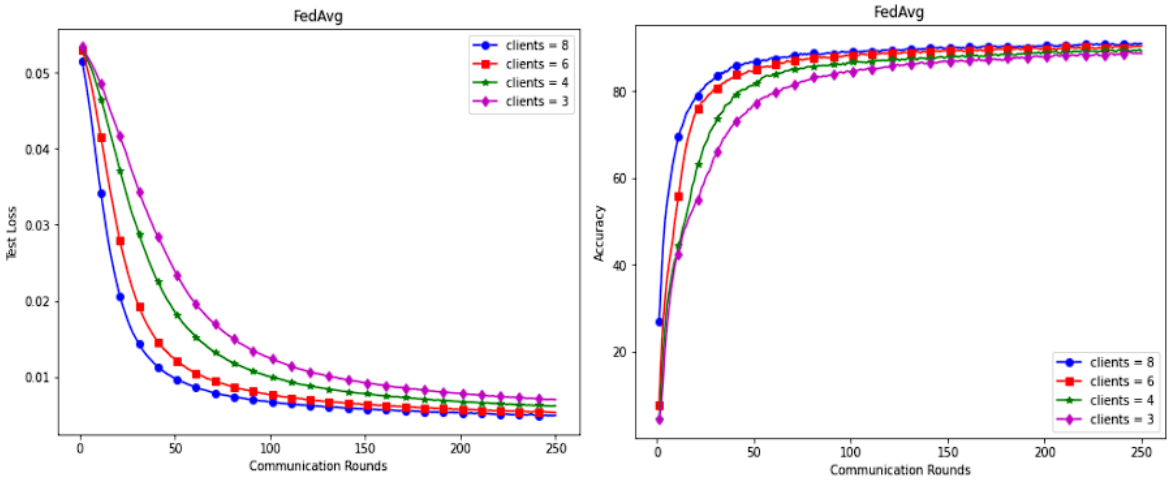

Figure 5.1: The graph shows how the accuracy of FedAvg by sampling different clients for training

have also benchmarked RingFed for our dataset. We played with two different hyperparameters, namely γ, which determines how much weight each client in a ring gives to

its adjacent client during pre-aggregation. We have experimented with different values of γ to find the optimal one. The best result is obtained when the γ value is 0.8 with an accuracy of 89.65% on our dataset, and the worst is obtained when γ is 0.5 with an accuracy of 88.96% . We have used 8 clients while playing around with γ. Detailed results are captured in Table 5.3.

5.2.2 Learning Rate (η) Experiments

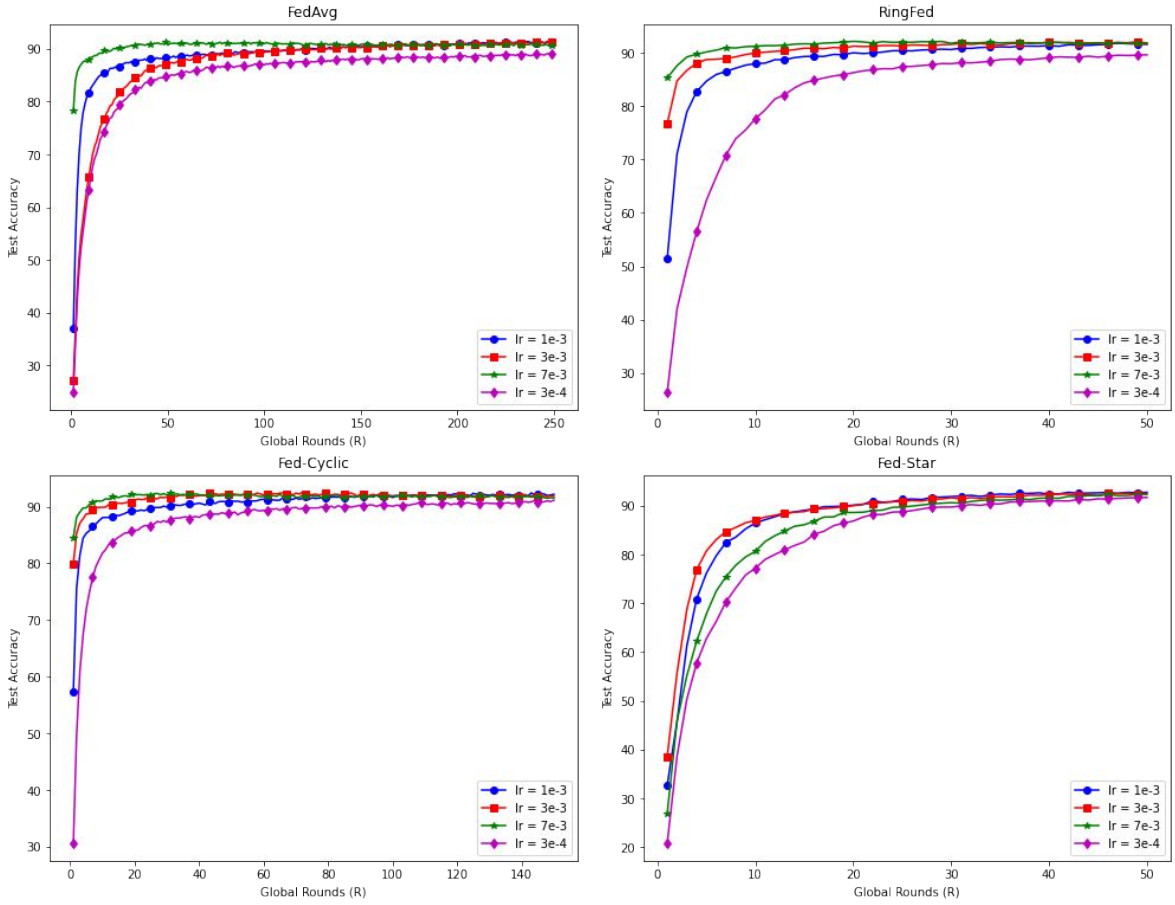

Figure 5.2: The graph shows how the accuracy of FedAvg, RingFed, Fed-Cyclic and Fed-Star changes with different learning rates (lr).

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

Learning Rate (η)	FedAvg [2]	RingFed [4]	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 5.7: The table shows the accuracy value after convergence attained by FedAvg, RingFed, Fed-Cyclic and Fed-Star for different values of learning rates.

rate, and maximum accuracy is obtained for the learning rate of 3e-3 with the value of 91.43%. For RingFed, we observed that 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 5.2 and Table 5.7.

5.2.3 Personalized Learning

In Table 5.8, 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 $E = 250$). FedAvg and RingFed do not show much personalization as their global models perform worse than local models. FedAvg performs poorly than those trained locally for 5 out of 8 clients. In contrast, RingFed performs poorly for 4 out of 8 clients and therefore fails to capture the statistical heterogeneity of the clients. Fed-Cyclic also performs better than the local clients for 6 out of 8 sources but performs marginally inferior on the Depositphotos and iStock datasets than the client trained locally. Fed-Star outperforms all the models trained locally and attains higher accuracy than all Federated algorithms evaluated on the different sources. We verified our hypothesis that Fed-Star captures the statistical heterogeneity and domain shift among the clients well, generates a global model fair to all local clients with different data distribution, and converges in fewer global rounds with higher accuracy.

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 5.8: Fed-Star outperforms all the local models trained using traditional ML method and baselines and Fed-Cyclic on different sources showing its personalization capabilities.

5.2.4 Non-IID (class label imbalance) data evaluation

We evaluate all three algorithms (FedAvg [2], Fed-Cyclic and Fed-Cyclic with Label Averaging) on CIFAR-10 dataset. We have divided the dataset in 80:20 for the train-test split and the train data is divided into 20 clients for FedAvg in which we selected 12 clients. For the Fed-Cyclic and Fed-Cyclic with label averaging, we selected 12 clients from the total of 12 clients. The results are summarized in table 5.9. Both of our methods perform better than the baseline with an average increase of more than 1% in classification accuracy.

Method	Accuracy	Weighted F1	Macro-F1
FedAvg	88.91%	88.80%	89.80%
Fed-Cyclic	90%	89.92%	89.92%
Fed-Cyclic w/ L.A.	90.35%	90.12%	90.08%

Table 5.9: Image classification result on CIFAR-10 for baseline and proposed methods. Here, red denotes the best value,blue denotes the second best value. Our method Fed-Cyclic in conjugation with Label Averaging performs better than the SOTA.

Chapter 6

Conclusion & Future Scope

6.1 Conclusion

Through our work, we accomplished multiple goals. We studied the collaboration of different commercial sources in a real-world setting using Federated Learning. We proposed 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 is artificially divided. We benchmarked it used to truly understand the application of Federated Learning in the field of computer vision in real-world scenarios. We successfully tackled the problem caused by domain shift among the clients due to heterogeneous data sources. We proposed two algorithms namely Fed-Cyclic and Fed-Star. Our algorithms have better convergence and better accuracy than the SOTA algorithms. Also, they perform much better from the personalization point of view, making them very relevant for meaningful collaboration among clients having statistical heterogeneity (domain shift). We validated our results on different datasets to show that our algorithm works well with generalisation.

In the second part of our work, we applied our algorithms to understand how they work with publicly available datasets to handle the real-world scenario of non-IID data due to a class-label imbalance among different clients. We combined Fed-Cyclic with label averaging to mitigate the issue of non-IIDness among the clients and performed better than SOTA.

6.2 Future Scope

For the initial part of the work, where we worked on the problem of domain shift, we can work by further extending our experimentation on different publicly available datasets. We can also include some more SOTA methods to further validate or improve our work.

For the second part of the work, we can further explore different techniques to overcome the issue of class-label imbalance since we explored only one single approach. We can also work on different publicly available datasets. We can also make further improvements by finding better techniques to tackle the problem of class-label imbalance by improving the proposed model and at the dataset level. We can also improvise our algorithms and compare them with other SOTA methods to identify the strength and weaknesses of our proposal correctly.

Bibliography

[1] S. K. Lo, Q. Lu, L. Zhu, H.-Y. Paik, X. Xu, and C. Wang, "Architectural patterns for the design of federated learning systems," *Journal of Systems and Software*, vol. 191, p. 111357, 2022.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[3] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.

[4] G. Yang, K. Mu, C. Song, Z. Yang, and T. Gong, "Ringfed: Reducing communication costs in federated learning on non-iid data," *arXiv preprint arXiv:2107.08873*, 2021.

[5] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *European conference on computer vision*. Springer, 2010, pp. 213–226.

[6] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[7] G. Drainakis, K. V. Katsaros, P. Pantazopoulos, V. Sourlas, and A. 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)*. IEEE, 2020, pp. 1–8.

[8] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[9] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, "A hybrid approach to privacy-preserving federated learning," in *Proceedings of the 12th ACM workshop on artificial intelligence and security*, 2019, pp. 1–11.

[10] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Computers & Industrial Engineering*, vol. 149, p. 106854, 2020.

[11] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, "Federated learning for internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1622–1658, 2021.

[12] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.

[13] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for on-device federated learning." 2019.

[14] J. Quinonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset shift in machine learning*. Mit Press, 2008.

[15] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern recognition*, vol. 45, no. 1, pp. 521–530, 2012.

[16] S. Zhang, A. E. Choromanska, and Y. LeCun, "Deep learning with elastic averaging sgd," *Advances in neural information processing systems*, vol. 28, 2015.

[17] P. Richtárik and M. Takáč, "Distributed coordinate descent method for learning with big data," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2657–2681, 2016.

[18] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, "Optimal distributed online prediction using mini-batches." *Journal of Machine Learning Research*, vol. 13, no. 1, 2012.

[19] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[20] M. Amiri-Zarandi, R. A. Dara, and E. Fraser, "A survey of machine learning-based solutions to protect privacy in the internet of things," *Computers & Security*, vol. 96, p. 101921, 2020.

[21] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," *arXiv preprint arXiv:1611.03814*, 2016.

[22] E. De Cristofaro, "An overview of privacy in machine learning," *arXiv preprint arXiv:2005.08679*, 2020.

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

[24] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," *Advances in neural information processing systems*, vol. 33, pp. 7611–7623, 2020.

[25] W. Chen, K. Bhardwaj, and R. Marculescu, "Fedmax: Mitigating activation divergence for accurate and communication-efficient federated learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2020, pp. 348–363.

[26] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," *arXiv preprint arXiv:2003.00295*, 2020.

[27] X. Liang, S. Shen, J. Liu, Z. Pan, E. Chen, and Y. Cheng, "Variance reduced local sgd with lower communication complexity," *arXiv preprint arXiv:1912.12844*, 2019.

[28] C. Chen, Z. Chen, Y. Zhou, and B. Kailkhura, "Fedcluster: Boosting the convergence of federated learning via cluster-cycling," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 5017–5026.

[29] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5132–5143.

[30] D. Chen, C. S. Hong, Y. Zha, Y. Zhang, X. Liu, and Z. Han, "Fedsvrg based communication efficient scheme for federated learning in mec networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 7, pp. 7300–7304, 2021.

[31] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *arXiv preprint arXiv:1811.11479*, 2018.

[32] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[33] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," *arXiv preprint arXiv:2002.10619*, 2020.

[34] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang, "Self-balancing federated learning with global imbalanced data in mobile systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 59–71, 2020.

[35] Q. Wu, X. Chen, Z. Zhou, and J. Zhang, "Fedhome: Cloud-edge based personalized federated learning for in-home health monitoring," *IEEE Transactions on Mobile Computing*, 2020.

[36] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1698–1707.

[37] Z. Chai, A. Ali, S. Zawad, S. Truex, A. Anwar, N. Baracaldo, Y. Zhou, H. Ludwig, F. Yan, and Y. Cheng, "Tifl: A tier-based federated learning system," in *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, 2020, pp. 125–136.

[38] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 8, pp. 3710–3722, 2020.

[39] M. Xie, G. Long, T. Shen, T. Zhou, X. Wang, J. Jiang, and C. Zhang, "Multi-center federated learning," *arXiv preprint arXiv:2005.01026*, 2020.

[40] Y. Deng, M. M. Kamani, and M. Mahdavi, "Adaptive personalized federated learning," *arXiv preprint arXiv:2003.13461*, 2020.

[41] C. Briggs, Z. Fan, and P. 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)*. IEEE, 2020, pp. 1–9.

[42] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19 586–19 597, 2020.

[43] L. Huang, A. L. Shea, H. Qian, A. Masurkar, H. Deng, and D. 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*, vol. 99, p. 103291, 2019.

[44] M. Duan, D. Liu, X. Ji, R. Liu, L. Liang, X. Chen, and Y. 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)*. IEEE, 2021, pp. 228–237.

[45] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[46] J. Luo, X. Wu, Y. Luo, A. Huang, Y. Huang, Y. Liu, and Q. Yang, "Real-world image datasets for federated learning," *arXiv preprint arXiv:1910.11089*, 2019.

[47] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[48] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[49] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.