2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)

FLaPS: Federated Learning and Privately Scaling

Sudipta Paul
School of Computer Sciences
National Institute of Science, Education and Research
Bhubaneswar, India
Homi Bhabha National Institute,
Anushaktinagar, Mumbai - 400094, India
sudiptapaulvixx@niser.ac.in
ORCID: 0000-0001-6561-997X

Poushali Sengupta
Department of Statistics
University of Kalyani
Kalyani, West Bengal, India - 741235
tua.poushalisengupta@gmail.com
ORCID: 0000-0002-6974-5794

Subhankar Mishra
School of Computer Sciences
National Institute of Science, Education and Research
Bhubaneswar, India
Homi Bhabha National Institute,
Anushaktinagar, Mumbai - 400094, India
smishra@niser.ac.in
ORCID: 0000-0002-9910-7291

2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS) | 978-1-7281-9866-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/MASS50613.2020.00011

***Abstract*—Federated learning (FL) is a distributed learning process where the model (weights and checkpoints) is transferred to the devices that posses data rather than the classical way of transferring and aggregating the data centrally. In this way, sensitive data does not leave the user devices. FL uses the FedAvg algorithm, which is trained in the iterative model averaging way, on the non-iid and unbalanced distributed data, without depending on the data quantity. Some issues with the FL are, 1) no scalability, as the model is iteratively trained over all the devices, which amplifies with device drops; 2) security and privacy trade-off of the learning process still not robust enough and 3) overall communication efficiency and the cost are higher. To mitigate these challenges we present Federated Learning and Privately Scaling (FLaPS) architecture, which improves scalability as well as the security and privacy of the system. The devices are grouped into clusters which further gives better privacy scaled turn around time to finish a round of training. Therefore, even if a device gets dropped in the middle of training, the whole process can be started again after a definite amount of time. The data and model both are communicated using differentially private reports with iterative shuffling which provides a better privacy-utility trade-off. We evaluated FLaPS on MNIST, CIFAR10, and TINY-IMAGENET-200 dataset using various CNN models. Experimental results prove FLaPS to be an improved, time and privacy scaled environment having better and comparable after-learning-parameters with respect to the central and FL models.**

***Index Terms*—Federated Learning(FL), Distributed Learning(DL), Differential Privacy(DP), Federated Averaging(FedAvg), k-means Clustering**

I. Introduction

Smartphones, Tablets, e-reader devices etc. have become an intrinsic part of our daily life with its different sophisticated features. According to a survey [17] the vast majority of Americans: 96% has an ownership of cellphone. In India, it is evaluated that by 2022 the smartphone users will extend to 442 million [16]. The powerful sensors that these devices consist of i.e. camera, GPS, microphone etc. have the power to generate and preserve data from different places whether it is "on the move with steady connection" or "idle and not under steady connection". From storing contact numbers, songs to important documents, doing banking transaction, insurance payment, taking and storing photos as a token of remembrance of the personal milestones etc.- these smart goodies have become an identity to its owner and have an access to an extraordinary amount of private and sensitive information about the real world. These data have a huge variation, distribution and are non-iid and sensitive to privacy attacks such as: DDOS, man in the middle attack, SQL injection, eavesdropping attack etc. These private and sensitive raw data are very useful for the improvement of different machine learning models [1] i.e. image processing models, speech to text generation, mobile keyboard spelling correction function for different languages, face recognition function etc.

Challenges The aforementioned machine learning models are normally trained in a centralized environment. But the sensitive nature of these smart devices' data is strongly against to store them centrally as it includes uploading them to a cloud based or hardware based central server. Even if the transaction is safe, there are possibilities that the information can be leaked at the time of the training too. Thus we need a robust privacy protection system, which can be scaled in every steps of the learning reasonably.

To solve these challenges, *Distribution Learning* [18](DL) for computation across multiple servers. The setbacks of DL observable from these works are: it is more focused on the power saving using parallel computing, the data is assumed to be iid - with almost same size at each server, the number of servers to be trained is limited, and all the servers need to be connected. Federated Learning [19], [20] was proposed to alleviate these problems. Here, the model itself goes to the data rather than data comes to be stored and trained in a central manner. We will know more about how FL works in the Section III. However, it also suffers from (1) Device drop, (2) Privacy traded off with the utility due to the lack of scalability in the usage of Differential Privacy (DP). Also no scalability mechanism has ever been pushed to the system of FL to prevent the utility-privacy trade-off while aggregating DP reports.

Contributions Acknowledging these challenges we are proposing a scalable FL system, **FLaPS** using a novel privacy

This research was partially supported by Department of Science and Technology Grant. Grant ID: NRDMS/UG/S.Mishra/Odisha/E-01/2018.

2155-6814/20/$31.00 ©2020 IEEE
DOI 10.1109/MASS50613.2020.00011

scaling method in four steps to elevate the normal FL. The contributions of our model are:

- Novel distributed learning framework **FLaPS** that improves upon FL in scalability and has comparable performance.
- Two privacy scaling steps at the cluster centres and two in the central server, gives novel privacy scaling mechanism.
- Scalable and tunable communication channels through cluster heads. E.g. normal FL needs to visit 2 million devices, but FLaPS needs to visit 100 - 2000 cluster centres only.

Section II describes the related works in this field, Section III describes the necessary background to build the base to understand our experiment, Section IV describes our proposed algorithm, Section V describes our experiment in details, Section VI discusses the result and analysis, and lastly Section VII discusses the future scope.

II. Related Work

Two main aims of FL according to Kairouz et al. [12] are maintaining the communication efficiency even if the network connection is not steady and maintaining the privacy and security of the sensitive data holders at the time of the model training. By communication efficiency we are aiming the situations where even after the device getting dropped, the turn around time will be lesser to go back again and after the updation of weights in the central server, the model quality is still good.

Various solutions has been discussed over time acknowledging them. Bonawitz et al. in their 2016 work [5] proposed a communication efficient secured aggregation protocol based on secret sharing, Konečný et al. in their 2017 work [2] proposed two solutions e.g. *structured updates:* where the model learns an update from a restricted space which is later parametrized with a smaller number of variables and sketched updates: where the model learns the whole update, then compress and send it to the server, for better communication efficiency. In their 2019 work [6] Caldas et al. initiates two new schemes to lower the communication costs: (1) they used lossy-compression on the global model at the time of transaction it from the server to the clients; and (2) a new technique, Federated Dropout, which enabled the users to train the global model efficiently at the local setup, using compact subsets of the model, which in turn gave a cutting in the communication and local computation cost. In their 2019 work [7] Bonawitz et al. presented a new model for secure and efficient communication. The proposed model auto-tune the aggregation, using particular characteristics of random rotation e.g. the foreseeable distribution of vectors after-rotation and the basic modular wrapping in secure aggregation. In their work [8] in 2019, Jiang et al. proposed that a model trained using a standard datacentre optimization method is much harder to personalize, compared to one trained using **FedAvg**. In their 2019 work [9], Sun et al. observed that in the absence of defences, the performance of the attacks largely depends on the fraction of adversaries present and the "complexity" of the targeted task. Moreover, they showed that *norm clipping and "weak" DP mitigate the attacks without hurting the overall performance*. In 2020, Reddi et al. [10]'s results give an insight on the interaction between client diversity and efficiency of communication . They plotted the federated varieties of adjustable optimizers, like ADAGRAD, ADAM, and YOGI. They inspect their convergence in the presence of diversified data for common nonconvex environment. In their 2020 work, Zhu et al. [11], based on big wheels in a sample of user-produced data flows, propose a distributed and privacy-preserving algorithm. They grasp the sampling and threshold properties of their algorithm to demonstrate that it is intrinsically obeys the rules of DP, without called for extra noise. None of these solutions ever highlighted the following points:

- The turn-around time bottleneck for each round when a device gets dropped.
- Utility and privacy, that are maintained through each step of the security maintenance mechanism.

These two shortcomings have mainly fueled us to propose the solution in the Section IV.

III. Background Theories and Inspiration

A. Federated Learning

The term "*Federated Learning*" or **FL** was first coined by McMahan et al. [1]. By their words - *"We term our approach FL, since the learning task is solved by a loose federation of participating devices (which we refer to as clients) which are coordinated by a central server."* The aim of FL is to learn a shared model by local aggregation. In this system the data is left in the devices and the central server learns the aggregation of the local weight updates after the training of the shared model through FedAvg algorithm. FedAvg combines local stochastic gradient descent (SGD) on each client at a central server that performs the weight averaging. The general objective is of the form:

$$\min_{w \in \mathbf{R}^d} f(w) \text{ where } F_k(w) = \frac{1}{n_k} \sum_{u \in P_k} f_i(w) \qquad (1)$$

where, we typically take $f_i(w) = l(x_i, y_i; w)$, that is, the loss of the prediction on example (x_i, y_i) made with model parameters w.

B. Differential Privacy(DP)

To satisfy individual privacy in a sensitive, statistical database, DP [3] has now become the standard. Here, we use two DP algorithms. The description of the algorithms are following: BUDS [14] or *Balancing utility and Differential privacy by shuffling* algorithm takes advantage of attribute reduction as well as iterative shuffling to ensure high utility while maintaining privacy. ARA [13] or *Aggregated RAPPOR and Analysis* is a centralized DP approach. In this system, the RAPPOR [21] reports were taken. Then a weighted sum is calculated using the ARA constants according to the positions on the report bit string which have been already identified

from using TF-IDF on the RAPPOR reports. The aggregated approach of ARA gave a satisfactory result which is a direct inspiration to use it at the central server and cluster centers to retrieve the weights and data respectively.

C. Clustering Algorithm

Cluster analysis is a technique by which a set of data-points, objects or entities gets divided into groups in such a way that those data-points, objects or entities are similar among themselves in some senses. We used here the Hartigan & Wong [4] variation of *k-means clustering* algorithm.

IV. FLaPS: Federated Learning and Privately Scaling

As we have learned the background theories, now we are describing our system in the following:

- Suppose, we have n numbers of user devices who agreed to be trained. Let the central server choose a number k ,where $2 < k < n$. All the n users will get clustered using *k-means clustering* algorithm depending on the budget of k. After this the k cluster centres will only have communication with the central server.
- The k cluster centres then send a message to the other cluster members to collect the differentially private data (produced using *BUDS* algorithm). The collected data will be then aggregated (using *ARA* algorithm).
- The k cluster centres will then train the already downloaded model from the central server with the aggregated data.
- After training, the weights and checkpoints of the trained model at each cluster centres will again get differentially privatized using the *BUDS* algorithm.
- The reports are then uploaded to the central server and the *ARA* algorithm then aggregates them to get a final report and update the central model according to the final report and the *FedAvg* algorithm.

The formal algorithm of the above description is given in Algorithm 1. The flow chart of the proposed algorithm's architecture is given in the Figure 1

Algorithm 1 FLaPS: Federated Learning and Privately Scaling

Input: n user devices ready to be trained with their data
Output: A centrally federated trained machine learning model

At the Server:
n idle users is selected
$k \in n$ gets randomly selected
for each $k \in n$ **do**
  $kmeans\_clustering(k, n)$
end for
At Each k Clusters:
n is the set of user devices
$C = C_1, C_2, C_3, ..., C_k$ is the set of clusters where $k \geq 2$
for each user $n_j \in C_i$ **do**
  $Report_j = BUDS(a, n_j)$, where a are attributes
  $Report_j$ is transmitted to the C_i centres
end for
for each $C_i \in C$, where each C_i is the server centres **do**
  Model downloaded from the central server
  $W_{agg} = ARA(Report_j, Reports)$
  Model gets trained with W_{agg}
  $Report_k = BUDS(w, C_k)$, where w is the weights of the trained model
  $Report_k$ is transmitted to the central server
end for
At the Server:
for each $Report_k \in Reports$ **do**
  $W = ARA(Report_k, Reports)$
end for
$W_{fed} = FedAvg(W)$
Testing Accuracy, Prediction Accuracy, AUC, F1 Score is calculated

TABLE I: Description of datasets

Dataset Name	Image Scale	Train	Test	Validation	Image Size	Class
MNIST	Grey	60000	10000	Null	28x28	10
CIFAR-10	RGB	50000	10000	Null	32x32	10
Tiny-ImageNet-200	RGB	100,000	10,000	10,000	64x64	200

V. Experiment

A. System Description

The system specification is - UBUNTU 19.04 64-bit Kernel Linux 5.0.0-37-generic x86_64 MATE 1.20.4 OS with 502.6 GB of memory and Intel Xeon(R) Gold 6138 CPU @ 2.00Ghz x 80, 4 GPU components(GeForce RTX 2080i) with CUDA version 10.1, driver version 430.50 with 6 TB external disk space. The R-studio with R-version 3.6.1, Keras(version 2.2.5.0) and Tensorflow(version 2.0.0) has been used, with a steady internet connection with a 8.70 Mbps download speed and a 11.96 Mbps upload speed.

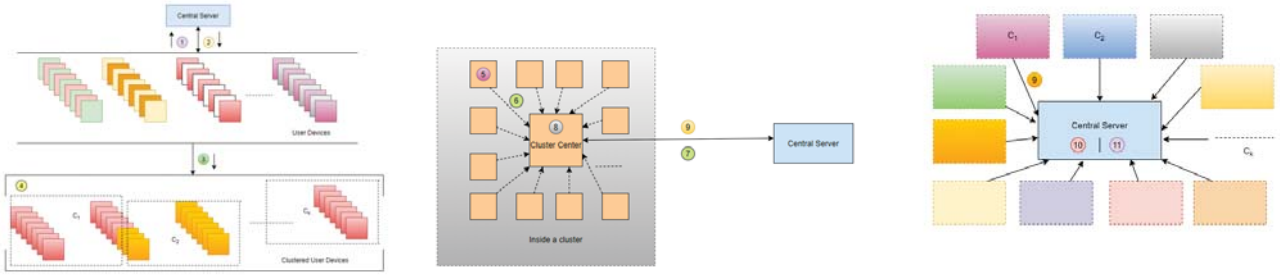
B. Learning Model and Data Description

We have considered CNN model for MNIST, CNN model for CIFAR10, InceptionResnetV2, MobilenetV2 as machine learning models for MNIST, CIFAR10 and ImageNet datasets.

C. Experiment Description

The experiment can be easily divided into re-usable subcomponents to write the program efficiently, i.e. central server, communication channels, clients and cluster formation. To impose a secure channel the *BUDS* algorithm has been implemented for generating DP reports. It has been used in two out of four communication steps in the proposed model. Also, the users are connected through a simple socket connection to the central server and the respective cluster centers. Find the codes here: github.com/smlab-niser/flaps. The experiment works like this:

1) The dataset is divided randomly among the clients. The



(a) At the server (b) At each k clusters (c) At the server

Fig. 1: The whole flow of FLaPS: (1) The User devices let know the Central Server that they are ready (2) The Central Server randomly choice a budget for k and send it to the users (3) The budget of k is broadcasted to the devices. (4) k Clusters formed with the help of the k-means algorithm.(5) Using BUDS algorithm the user data is differentially privatizing. (6) The DP report from each user is collecting at the cluster centre. (7) The training model is downloaded from the central serve.r (8) The model is trained using the ARA aggregated user report. (9) The weights of the model produces a DP report and is uploaded to the central server.(10) The reports are aggregated using ARA. (11) The weights are updated using FedAvg Algorithm

attributes that each client has to maintain are their user-ID, Number of images they have been allotted, Cluster-ID, and the maximum and minimum Indexes of the allotted images. The value of Cluster-ID gets changed every time according to the central server demand.

2) The value of these attributes are one-hot encoded and saved at the individual client.
3) The central server tries to find which client devices are idle or ready to train a machine learning model at this stage. After finding those client devices the central server then randomly choose the budget of k. With respect to the budget of the k, the clusters formed among the user devices using the *k-means clustering algorithm*.
   For our experimental purpose 200 clients has been chosen and the budget of k is being varied from 2-20.
4) After the cluster formation happened, the cluster centres of the k clusters send a message to the other clients in the clusters to find the range of the image indexes they possessed individually. The max index and the min index attribute gets together to form a single attribute at each client side and then with the help of a shuffler the values of each attribute get shuffled inside a cluster. After shuffling, statistics of the indexes left as it was but the user-IDs and the number of images per clients get shuffled. Therefore, a differentially private report is being generated here.
5) After getting the reports through a socket connection, an aggregation step to get full insight from all these reports takes place. The *ARA* architecture has been used here. This aggregation happens at all the k cluster centres for its respective clusters. The k clients then download the model from the central server and train the model with the help of the statistics it gained from the aggregated differentially private reports.
6) After the training step the weights are then extracted and differentially privatized using *BUDS* and the information loss is calculated. When the loss converges to a certain value the report of the weights are then sent to the central server.
7) After getting the reports from all the k clusters the central server then aggregates them using *ARA* with a loss optimization. According to that aggregation, the central server then replaces the weights with the aggregated(normally averaged) weights.

VI. Result and Analysis

A. Complexity Analysis

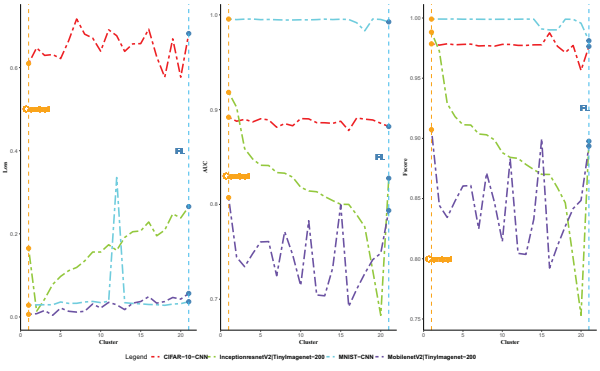
We have divided our framework into 4 distinct measurable time steps for complexity as well as performance analysis. The time division, it's description and complexity analysis of the proposed FLaPS algorithm for different time division is in Table II.

TABLE II: Complexity analysis of FLaPS

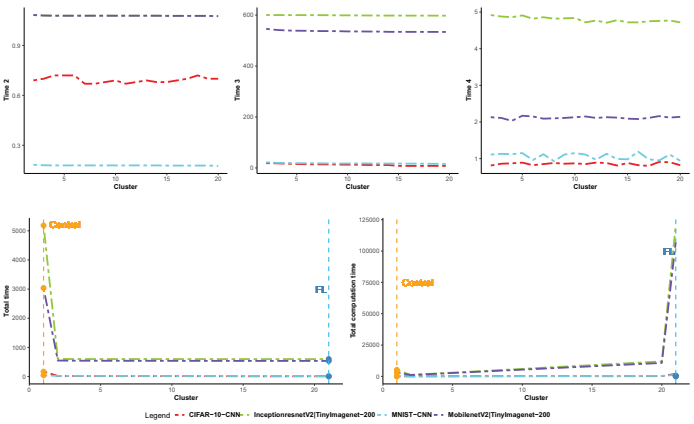
Time	Description	Complexity
Time 1	Random user choice and cluster formation in average time and datasets	$\mathcal{O}(n^2)$
Time 2	Generating differentially private report from the clustered users and sending those reports to the randomly chosen cluster centres in average time	$\mathcal{O}(n+v^2)$
Time 3	Average training time per cluster and generating differentially private report of the weights + Training per cluster [15]	$\mathcal{O}(dc)$
Time 4	Recieving the differentially private reports of weights from each cluster in average plus the federated average algorithm run time	$\mathcal{O}(c^2+dc)$

B. Empirical Analysis

The results for all the experiments are arranged in the Tables IV and III. The plotted graphs using these results are in Figures 2b and 2a. The empirical analysis can be done from two perspectives: 1) analysing the after training parameters, 2)



(a) The Loss, AUC and Fscore comparison for all the architectures in FLaPS, Central and FL environment. The orange dot is for the central time and the steel blue dot is for the Federated learning time in the same environment.



(b) Time comparison across central, FL and FLaPS.

TABLE III: Loss, AUC and f-score data for all four architectures

	MNIST-CNN			CIFAR-10-CNN			Tiny-ImageNet-200-MobinetV2			Tiny-ImageNet-200-InceptionresnetV2		
Cluster	loss	AUC	f-score	loss	AUC	f-score	loss	AUC	f-score	loss	AUC	f-score
2	0.029106	0.995	0.99912	0.647612	0.8878	0.9775	0.007813	0.744363	0.844363	0.013672	0.90248	0.97248
3	0.029368	0.9955	0.9991	0.629345	0.88978	0.97856	0.015625	0.734335	0.834335	0.042969	0.858893	0.928893
4	0.029309	0.9956	0.9992	0.63146	0.88705	0.977741	0.003906	0.748025	0.848025	0.078125	0.848004	0.918004
5	0.03606	0.99494	0.99898	0.621674	0.890389	0.978078	0.021484	0.760447	0.860447	0.097656	0.841321	0.911321
6	0.03266	0.995167	0.99903	0.664708	0.889167	0.9783	0.013672	0.760812	0.860812	0.111328	0.841079	0.911079
7	0.032855	0.994834	0.998967	0.717057	0.88144	0.97678	0.011719	0.724809	0.824809	0.119141	0.833645	0.903645
8	0.036057	0.99456	0.998911	0.680497	0.885278	0.977056	0.013672	0.771309	0.871309	0.134766	0.832965	0.902965
9	0.03735	0.994611	0.998922	0.671125	0.883	0.9766	0.03125	0.746856	0.846856	0.15625	0.828764	0.898764
10	0.034552	0.99483	0.998967	0.639487	0.89056	0.9781	0.021484	0.714381	0.814381	0.15625	0.818144	0.888144
11	0.037673	0.99472	0.99894	0.690932	0.890278	0.97805	0.035156	0.782813	0.882813	0.173828	0.814217	0.884217
12	0.33553	0.995	0.999	0.676812	0.886278	0.97725	0.029297	0.704479	0.804479	0.160156	0.813392	0.883392
13	0.03434	0.9955	0.9991	0.639058	0.886167	0.97723	0.017578	0.703596	0.803596	0.191406	0.808335	0.878335
14	0.032519	0.995056	0.999012	0.657092	0.88555	0.9777	0.033203	0.732946	0.832946	0.205078	0.804059	0.874059
15	0.031592	0.9954	0.99089	0.657693	0.8878	0.97775	0.037109	0.799668	0.899668	0.207031	0.800149	0.870149
16	0.03	0.995123	0.990118	0.6926	0.878	0.987655	0.048828	0.692272	0.792272	0.228516	0.800036	0.870036
17	0.0295	0.991654	0.99014	0.626	0.891238	0.97654	0.033203	0.710671	0.810671	0.195313	0.789063	0.859063
18	0.028	0.983195	0.999	0.576926	0.889988	0.971321	0.037109	0.727026	0.827026	0.208984	0.776445	0.846445
19	0.031	0.995678	0.9989	0.66926	0.889123	0.977163	0.046875	0.741714	0.841714	0.248047	0.730246	0.800246
20	0.032	0.995134	0.995689	0.576926	0.885643	0.956432	0.042969	0.74847	0.84847	0.238281	0.683008	0.753008
FL	0.0367	0.9926	0.9811	0.68201	0.8822	0.9764	0.056641	0.793604	0.893604	0.265625	0.82765	0.89765
Central	0.0283	0.9956	0.99911	0.610001	0.892	0.9785	0.006344	0.80726	0.90726	0.165234	0.91825	0.98825

analysing the time consumption of each and every steps of the algorithm.

1) Model quality analysis after training: Three parameters: loss, AUC and Fscore are being taken here to analyze the after training model quality. Here, at the time of training: rmsprop is being used as the optimizer function and categorical crossentropy is being used as the loss function. All the result data is given in Table III. From the loss graph in figure 2a it is evident that the more non-iid and varied the data is, the loss is lesser or equal to the central learning. In case of 2-clustered FLaPS approach, the claim is always true irrespective of architectures and datasets. AUC measures the quality of the model's predictions irrespective of what classification threshold is chosen. For all the dataset i.e. MNIST, CIFAR-10 and TINY-IMAGENET-200 the proposed clustering approach has produced higher or almost equal AUC in comparison to the FL, which in turn proves that the quality of the learned model is not degrading at all after our training approach. All the result data are in the Table III. From the graphs at Figure 2a it is evident that in case of MNIST-CNN and CIFAR-10-CNN model the Fluctuations of AUC score is almost linear in comparison to the central and the FL models. From the mobilenet and inceptionresnet model AUC score graph it is evident that more the non iid data the model learns better. The Fscore, is a measure of a test's accuracy. More the value of Fscore, better the model. The tendency of learning in the Fscore graphs 2a is also looking same as discussed in the previous paragraph regarding the AUC graphs.

2) Time analysis: For the second approach of analysis, the CPU clock has been used to measure the run-time. All the time is measured in seconds. The Time 1 division is almost same for all the experiments. That is why we did not plot. The Time 2 division was same for both the architectures for InceptionresnetV2 and MobilenetV2, this is why only the MobilenetV2 curve is showing in the plot. Analysis of plots reveal that most of the time is taken by the Time 3 step

TABLE IV: Time data for all four architectures

		MNIST-CNN				CIFAR10_CNN				Tiny-ImageNet-200-MobileNetV2				Tiny-ImageNet-200-InceptionresnetV2			
cluster	t1	t2	t3	t4	tt	t2	t3	t4	tt	t2	t3	t4	tt	t2	t3	t4	tt
2	0.083	0.183	22.14	1.11	23.516	0.69	18.79	0.81	20.76	1.083	545.91	2.13	548.813	1.083	600.21	4.92	600.98
3	0.08	0.18	20.11	1.13	21.5	0.7	18.15	0.86	20.17	1.08	541.82	2.11	544.71	1.08	600.03	4.88	600.81
4	0.079	0.179	19.24	1.12	20.618	0.72	16.82	0.87	18.85	1.079	539.71	2.03	542.539	1.079	599.93	4.86	600.73
5	0.079	0.179	18.74	1.15	20.148	0.72	15.51	0.89	17.56	1.079	538.62	2.17	541.589	1.079	599.89	4.91	600.69
6	0.079	0.179	18.41	0.96	19.628	0.72	14.62	0.82	16.6	1.079	538.3	2.15	541.249	1.079	599.8	4.82	600.6
7	0.079	0.179	18.2	1.12	19.578	0.67	14.2	0.85	16.21	1.079	537.29	2.09	540.129	1.079	599.83	4.86	600.58
8	0.079	0.179	17.76	0.93	18.948	0.67	13.88	0.88	15.92	1.079	537.18	2.1	540.029	1.079	599.53	4.82	600.28
9	0.079	0.179	17.56	1.11	18.928	0.68	13.32	0.86	15.34	1.079	536.95	2.11	539.819	1.079	599.13	4.83	599.89
10	0.079	0.179	17.62	1.15	19.028	0.69	12.97	0.87	15	1.079	535.89	2.13	538.789	1.079	598.99	4.84	599.76
11	0.079	0.179	17.76	1.11	19.128	0.67	12.65	0.85	14.66	1.079	535.73	2.15	538.629	1.079	598.72	4.72	599.47
12	0.079	0.179	17.67	0.97	18.898	0.68	11.67	0.89	13.72	1.079	535.51	2.11	538.379	1.079	598.5	4.77	599.26
13	0.079	0.179	17.45	1.13	18.838	0.69	11.58	0.88	13.62	1.079	535.29	2.13	538.189	1.079	598.23	4.71	598.99
14	0.079	0.179	17.23	0.99	18.478	0.68	11.43	0.81	13.4	1.079	534.22	2.12	537.099	1.079	598.02	4.78	598.78
15	0.078	0.178	17.26	0.98	18.496	0.68	8.09	0.88	10.13	1.078	534.22	2.09	537.068	1.078	597.99	4.72	598.75
16	0.078	0.178	17	1.19	18.446	0.69	8.03	0.82	10.01	1.078	534.12	2.08	536.968	1.078	597.91	4.72	598.68
17	0.078	0.178	16.89	0.98	18.126	0.7	7.99	0.8	9.95	1.078	533.99	2.11	536.878	1.078	597.9	4.75	598.68
18	0.078	0.178	16.35	0.95	17.556	0.72	8.06	0.91	10.13	1.078	533.95	2.16	536.908	1.078	597.88	4.76	598.68
19	0.078	0.178	16.31	1.11	17.676	0.7	8.1	0.9	10.16	1.078	533.99	2.12	536.888	1.078	597.78	4.77	598.56
20	0.076	0.176	15.14	0.94	16.332	0.7	7.98	0.82	9.95	1.076	533.63	2.14	536.546	1.076	597.41	4.72	598.2
FL			5.37	0.95	6.32		9.27	0.88	10.15		533.3	2.11	535.41		597.22	4.82	602.04
Central					41.14				169.69				3036.48				5182.44

or the training phase. Central learning takes the most total time and clustered and FL approach take almost similar or comparable time. If we sum up the Time 1 and Time 2 for every experiments the time overhead is almost 2 seconds. These summed up time measures the time limit of the data gathering at the cluster center. If any device gets lost, at this point, we can always look back to inspect but the training won't stop as there is data to provide, which solves the device drop problem. We can also see that more the number of weights, the Time 4 tends to get larger.

VII. Future Scope and Conclusion

FLaPS is a novel distributed (federated) learning framework where the training is leveraged at a few cluster heads rather than all the devices individually. It builds on the foundations set by FL and DP built keeping user privacy in mind. Our approach in FLaPS gives a better privacy and utility guarantee by the introduction of DP in combination with FL in four simple steps. Scalability is added by introducing clustering approach at the beginning before the training. Following the lesser number of devices acting as cluster heads; robustness is improved with quicker re-initiation through smaller turn around time. Therefore in conclusion FLaPS builds a bridge between the central and FL with better robustness, privacy and scalability. In the future we would like to explore other variations in different phases of the system such as clustering, data aggregation, faster as well as with different privacy settings. We would also like to test it on real mobile devices by setting up a test bed in our university.

References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data", arXiv preprint arXiv:1602.05629 , 2017.

[2] J. Konečný, H. B. McMahan, F. X. Yu, A. T. Suresh, D. Bacon and Peter Richtárik, "Federated Learning: Strategies For Improving Communication Efficiency", arXiv preprint arXiv:1610.05492, 2017.

[3] C. Dwork and A. Roth, "The algorithmic foundations of DP", Foundations and Trends® in Theoretical Computer Science, 9(3–4), pp.211-407.

[4] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm", Journal of the Royal Statistical Society. Series C (Applied Statistics). 1979 Jan 1;28(1):100-8.

[5] K. Bonawitz , V. Ivanov , B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel , ... & K. Seth," Practical secure aggregation for Federated Learning on user-held data." arXiv preprint arXiv:1611.04482.(2016).

[6] S. Caldas , J. Konečný, H. B. McMahan & A. Talwalkar. " Expanding the reach of Federated Learning by reducing client resource requirements". arXiv preprint arXiv:1812.07210.2019

[7] K. Bonawitz., F. Salehi, J. Konečný, H. B. McMahan, & M. Gruteser, "Federated Learning with autotuned communication-efficient secure aggregation." arXiv preprint arXiv:1912.00131.2019

[8] Y. Jiang , J. Konečný, K. Rush, & S. Kannan, "Improving Federated Learning personalization via model agnostic meta learning." arXiv preprint arXiv:1909.12488.2019

[9] Z. Sun, P. Kairouz, A. T. Suresh, & H. B. McMahan, " Can You Really Backdoor Federated Learning?." arXiv preprint arXiv:1911.07963.2019

[10] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush,J. Konečný, ... & H. B. McMahan," Adaptive Federated Optimization." arXiv preprint arXiv:2003.00295.2020

[11] W. Zhu, P. Kairouz, H. Sun,H.B. McMahan, & W. Li, " Federated heavy hitters discovery with DP." arXiv preprint arXiv:1902.08534.2020

[12] P. Kairouz, B. H. McMahan, B. Avent, A. Bellet, M. Bennis,N. A. Bhagoji, ... & G. R. d'Oliveira," Advances and open problems in Federated Learning." arXiv preprint arXiv:1912.04977.2019

[13] S. Paul, & S. Mishra," ARA: Aggregated RAPPOR and Analysis for Centralized DP", SN Computer Science, 1(1), 22.2020

[14] P. Sengupta, S. Paul & S. Mishra, "BUDS: Balancing Utility and DP by Shuffling", 11th Internation Conference on Computing, Communication & Networking Technologies. 2020

[15] K. He, & J. Sun, "Convolutional neural networks at constrained time cost". In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5353-5360).2015

[16] Statista, & Ms. M.Jaganmohan, (2020, May 7). Smartphone users in India 2015-2022 Published by Madhumitha Jaganmohan, May 7, 2020. Retrieved May 19, 2020, from https://www.statista.com/statistics/467163/forecast-of-smartphone-users-in-india/

[17] Pew Research centre: Internet, Science & Tech. (2019). Demographics of Mobile Device Ownership and Adoption in the United States. Re-

trieved May 18, 2020, from https://www.pewresearch.org/internet/fact-sheet/mobile/
[18] I. Diakonikolas, E.Grigorescu, J. Li, A. Natarajan, K. Onak, & L. Schmidt," Communication-efficient distributed learning of discrete distributions." In Advances in Neural Information Processing Systems (pp. 6391-6401).2017
[19] A. Hard, A. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, ... & D. Ramage. "Federated Learning for mobile keyboard prediction." arXiv preprint arXiv:1811.03604.2019
[20] M. Chen, R. Mathews, T. Ouyang, & F. Beaufays, "Federated Learning of out-of-vocabulary words." arXiv preprint arXiv:1903.10635.2019
[21] U. Erlingsson, V. Pihur and A. Korolova, November." Rappor: Randomized aggregatable privacy-preserving ordinal response". In Proceedings of the 2014 ACM SIGSAC conference on computer and communications security (2014) (pp. 1054-1067).