UPPSALA
UNIVERSITET

Federated Learning for Bioimage Classification

Jiarong Liang

Degree project in bioinformatics, 2020
Examensarbete i bioinformatik 30 hp till masterexamen, 2020
Biology Education Centre and Department of Information Technology, Uppsala University
Supervisor: Andreas Hellander

Abstract

Machine learning, especially deep learning, is becoming popular in biological and medical fields. However, even though a large amount of data is produced within these fields, due to privacy issues, the implementation of machine learning still faces the problem of lacking suitable datasets. A new machine learning setting called Federated learning is considered to have the ability to utilize small and private datasets without the leakage in privacy, providing more potential opportunities for data scientists.

In this project, we simulate federated learning under various scenarios using the CIFAR10 dataset, including model and data poisoning, client size, asynchronous node updating, sharing a public dataset. The simulation system can be edited for other scenarios as well. We also provide an example of a federated diabetic retinopathy detection task on the cloud. We hope this work can be a reference for bioinformaticians who want to attempt federated settings in their research.

Learning without sharing

Popular Science Summary

Jiarong Liang

Nowadays, biological or medical researchers have produced a large amount of data, which seems to be a perfect application field for machine learning. Machine learning, especially deep learning is becoming more and more popular in biological fields. In many cases, the researcher will ’teach’ a computer-based model to handle certain tasks using large and suitable datasets. These datasets can be found in various public databases. Many interesting attempts have already been done in applications such as developing drugs, assisting doctors, predicting epidemics, and so on.

However, in many other cases, researches are hindered by lacking suitable datasets, since biological or medical records sometimes include sensitive information and are not allowed to be accessed by researchers. For example, patients’ medical records from the hospitals should not be visited by anyone else except for the doctors. In another example, the gene sequence of the individuals are client’s genetic privacy, sequencing companies are not allowed to share them with genetic labs. In other situations where private information can be deduced from the records, publicizing can also be dangerous and illegal.

We all know that for complex tasks, feeding in more data to the model can usually result in better performance. Unfortunately, a suitable amount of data is not always available in a single institution, especially for the small ones. Since we cannot share these data, the datasets from small instances become somehow ’wasted’.

A new method has been developed for avoiding such limitations in data usage. This method, which is called Federated learning, was published in 2016 and has already been used in the development of several applications such as the Google keyboard. This method can make good use of the client’s private datasets without having access to them. In this project, we will test this method’s performance under several possible scenarios, and present an example of using this technique for disease detection.

Degree project in bioinformatics, 2020
Examensarbete i bioinformatik 30 hp till masterexamen, 2020
Biology Education Centre and Department of Information Technology, Uppsala University
Supervisor: Andreas Hellander

Table of Contents

Abbreviations

CNN	Convolutional Neural Networks
DR	Diabetic Retinopathy
FedAvg	Federated Averaging Algorithm
FedSGD	Federated Stochastic Gradient Descent Algorithm
LSTM	Long Short-Term Memory
HTTP/2	Hypertext Transfer Protocol Version 2
IID	Independent and Identically Distributed
RPC	Remote Procedure Call
SMC	Secure Multiparty Computation
TCP	Transmission Control Protocol

Software Links

TensorFlow	https://www.tensorflow.org/
gRPC	https://grpc.io/

Project GitHub Repository

https://github.com/Jiarong000/Thesis2020

1 Introduction

Machine learning methods have been applied to medical or biological researches for a long time. Traditional algorithms such as PCA, random forest, AdaBoost has been widely used (Benitez *et al.* 2011, Chen & Ishwaran 2012). In recent years, a subfield called deep learning is becoming popular. Traditional machine learning workflow requires manual preprocessing and feature extraction, while deep learning can learn features from raw data. (Angermueller *et al.* 2016) and (Jurtz *et al.* 2017) summarize deep learning's application in regulatory genomics, biological image and sequence analysis. Besides these, many interesting works have been done in recent years. DeepVariant aims to detect genetic variants from sequencing data (Poplin *et al.* 2018). DeepSimulator attempts to generate some more realistic base calls for simulation experiments (Li Y *et al.* 2018). Autoencoder has also been applied for denoising scRNA-seq datasets (Eraslan *et al.* 2019). Though the results of machine learning still require careful assessments from professionals in the application fields, it has a huge advantage in data mining and assisting professionals.

There is a consensus that a large dataset is required for deep learning tasks, especially when the task is complex. ChestX-ray14 is the largest dataset of chest radiographs, which contains 112120 images from 30805 unique patients. CheXNeXt is trained over this dataset and claims to approximate expert's performance (Rajpurkar *et al.* 2018). In another research, several structurally distinct antibacterial molecules have been discovered with the assistance of a model trained with an empirical dataset of size 2335 (Stokes *et al.* 2020). In many other cases, a suitable dataset can be collected from biological databases such as NCBI and EMBL or other specific databases such as TCGA and ICGC.

However, it is not always possible to collect an appropriate dataset. In many cases, datasets are small and unbalanced. In other cases, annotations are not suitable for the specific training task. Many institutes, for example, hospitals or sequencing companies, own a private dataset of their clients and have the ability to perform annotation. Due to privacy concerns and restrictions, this part of data is usually not allowed to be shared with other institutes or to the public, resulting in many small and isolated datasets.

A new machine learning setting called Federated learning is considered to have the potential of utilizing isolated datasets without leakage in personal privacy. In this project, we test federated learning's behavior under various scenarios and perform an example federated medical image classification task on the cloud. We hope this work can provide references for future federated designs and we provide a trivial example for understanding and building up a federated system.

There are several challenges in federated learning, including communication costs, security protection, statistical and system heterogeneity (Li T *et al.* 2019b). In this project, we do not put too much attention on communication or security issues.

2 Background

2.1 Federated Learning

Federated learning is a machine learning strategy where many clients can collaboratively train a model when their local dataset is inaccessible to each other. The system will be orchestrated by a central server, in which training results from clients are aggregated. Compared to centralized learning, it requires less storage or computational resources in the central server and most importantly, protects each client's private data.

There are many domains in federated learning. Cross-device federated learning is implemented on a large number of small devices while cross-silo learning is for collaboration among institutions (Kairouz *et al.* 2019). From another aspect, in most training cases there will be horizontal federated learning where client datasets are partitioned by samples and share similar feature space. In this case, gradients or weights are aggregated. On the other hand, client datasets only share user space in vertical learning, in which intermediate results such as hidden layer's representations are shared(Liu *et al.* 2018). Federated transfer-learning, applied when the intersection in both feature and user space is small, is also possible (Liu *et al.* 2018).

In most biological or medical use cases, cross-silo and horizontal federated learning will be performed among institutions. A successful case of brain tumor segmentation in federated settings has been reported (Sheller *et al.* 2018). However, federated settings are not popular in current biological researches.

2.2 Simple Federated Process

When we are training a machine learning model, we aim to use its function later, but the model should not be used for any purpose until it is evaluated and considered to be well-trained. Hence, a federated process contains two stages, the training stage and the application stage.

The training stage mainly contains several basic steps (Kairouz *et al.* 2019). The server will select clients who meet the requirement and broadcast model weights to the client. Then clients train the model with their local data and upload the information (i.e. weights, gradients, intermediate layers) to be aggregated. These steps will be repeated until training is

finished. Additional techniques and methods are applied to control these steps and encrypt shared messages.

The application phase is similar to that in central machine learning. The model will be published or kept private according to the agreement with the clients.

2.3 Federated Algorithms

Federated Averaging (FedAvg) and Federated SGD (FedSGD) are two popular algorithms in federated learning and many optimization algorithms are developed from them. In FedAvg settings, client weights will be collected and averaged while in FedSGD settings, gradients will be collected (Mcmahan *et al.* 2016).

Many other optimization algorithms are developed for various purposes, (Kairouz *et al.* 2019) provides a summary of popular ones, such as FedProx (Li T *et al.* 2018) and SCAFFOLD (Karimireddy *et al.* 2019), using either proximal term or control variates to reduce the effects caused by heterogeneous clients. This means, for example, weights are updated towards minimizing the loss function (f) in FedAvg. In FedProx, weights are updated to minimize ($f + u||w-W^t||^2$) in which a proximal term represents the distance between client weights (w) and global weights (W^t) is added in as a restriction factor. Several other methods such as FedMA (Wang *et al.* 2020) and federated transfer learning (Liu *et al.* 2018) share hidden elements instead of gradients or weights.

Below is the pseudo-code of FedAvg, which is used in this project. In each client node, gradient descent is used to minimize the loss function f, where η is the learning rate to control the speed of updates. The server collects updated weights from each client and generates the global weights for that round.

Server executes:
  Initialize W_0
  For each round t = 1, 2,..., T **do**
    For each client index by k= 1, 2,..., K **in parallel do**
      $W_{t+1}^k \leftarrow$ ClientUpdate(k,W_t)
    $W_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} W_{t+1}^k$
ClientUpdate(k,W):
  $B_k \leftarrow$ (local dataset of the k^{th} client in batches)
  For local step s = 1,..., S **do**
    For batch b $\in B_k$ **do**
      $W \leftarrow W - \eta \nabla f(W,b)$
  Return W to server

Algorithm 1. FedAvg

2.4 Previous Work

Many pieces of research are implemented on current challenges in federated learning includes communication cost, security protection, statistical and system heterogeneity (Li T *et al.* 2019b). The communication cost is usually a bottleneck for cross-device learning. To save the cost, compression schemes and multiple local updating can be introduced to the settings (Li T *et al.* 2019b). Privacy issues, such as retrieving information from two successive models, have solutions such as SMC, differential privacy and homomorphic encryption. These solutions are used in popular federated frameworks that support both deep learning and traditional machine learning. Other techniques such as blockchain, can also be integrated to make the system more secure (Passeratpalmbach *et al.* 2019).

Statistical heterogeneity (such as different sample distributions among nodes) can also cause performance degradation. Several methods to solve this issue aim to make the local training set less skewed, including sharing a small set of data (Zhao *et al.* 2018) and augmenting local set towards IID (Jeong *et al.* 2018). Some other methods, such as FedProx (Li T *et al.* 2018) add restrict factors to optimization function. FedNAS (He Chaoyang *et al.* 2020) and FedMA (Wang *et al.* 2020) also claims to reduce issues of non-IID data distribution.

System heterogeneity is also common. In some scenarios, such as in cross-device learning, since it is not a usual case to wait for all devices to be ready for training, the system will train on only the available devices at each time. In other cases, nodes will not update in synchronous ways if some nodes are too slow to be waited for, or nodes can be added in or drop out from the system during the training process. These are compromises for real-world applications and probably can cause the system to act in slower ways than identical. However, we are still uncertain about the extent of this impact.

Surprisingly, even with multiple optimization methods for these issues available, we cannot summarize an overview of statistical or system heterogeneity's impact on federated learning. Those impacts are tested mostly in limited scenarios as a comparison to the optimization methods. In different researches, the author may also use different datasets and evaluation standards. In (Kairouz *et al.* 2019) the authors purpose the mathematical methods to calculate convergence rated for non-IID datasets, however, only for ideal cases and is just theoretical. This fact can make many people stay uncertain about the extent of each issue's impact on their federated system. Intuition is important for designing a system, especially when it is impossible to make optimization in every aspect. In this project, we test several common issues in federated learning thoroughly, under several IID or non-IID circumstances, and with different amounts of data. We hope this empirical work can be a clear and easy-to-understand reference for those who need to implement federated learning in the future.

2.5 Federated Frameworks

In the Appendix of (Kairouz *et al.* 2019), several popular federated frameworks are summarized. TensorFlow Federated and PySyft provide tools for federated learning based on the widely used TensorFlow and PyTorch software. A platform called PyGrid is also under development, using PySyft as its basic library. However, by the end of May 2020, their official versions have not yet been released, only several simulation functions are available.

Another software called Federated AI Technology Enabler (FATE) has been released in 2019. FATE already have enabled several functions in both traditional learning and deep learning. Meanwhile, more functions are under construction.

2.6 Communication tools

In a federated system, clients need to communicate with the server. Browsing their source code, TensorFlow Federated seems to use gRPC and an example in PySyft uses WebSocket. gRPC is a high-performance RPC framework with HTTP/2-based transport. WebSocket provides computer communication over a single TCP connection. In this project, we use gRPC for communication.

2.7 Future in Biological Fields

As tools become more sophisticated and convenient, using machine learning techniques is becoming more and more convenient. Automated machine learning aims to assist non-experts to apply machine learning techniques in many aspects (Yao *et al.* 2018). Google's AutoML (Real *et al.* 2020), AutoAugment (Cubuk *et al.* 2018), Swish (Ramachandran *et al.* 2017) are popular examples.

At the same time, federated frameworks such as FATE can provide secure federating experience for their users. There are already attempts to apply AutoML in a federated way, typically the federated Neural Architecture Search (He Chaoyang *et al.* 2020, Xu *et al.* 2020). In the foreseeable future, collaboratively training a model with data from different institutes might no longer be a difficult task for biological researchers.

3 Dataset and methods

In this project, the CIFAR-10 dataset (Krizhevsky 2012) has been used for locally simulating different scenarios of federated learning and OIA-DDR dataset (Li T *et al.* 2019a) has been used for implementing an example medical image classification task on the cloud. The models are not the state of the art, but they are sufficient for testing scenarios for federated learning without loss of generality.

Most simulation tasks are tested for 1000 epochs (for centralized training) or global rounds (for federated learning) and the cloud implementations are tested for 100 epochs/rounds. The code is written in Python3 and TensorFlow 2.0.1 is used. Due to the limit of time and computational resources, tests are not repeated.

3.1 Numerical Experiments

Different scenarios in federated learning are simulated in this part. CIFAR-10 dataset is used for simulation experiments. The task for simulation is to input images of size 32x32x3 and classify them into 10 categories. The model is provided by Mattias Åkesson from Scaleout Systems. We apply the ADAM optimizer with the learning rate set to default (1e-3). The batch size is set to 100 (or below 100 in tests with 4k dataset, depending on available computational resources); the loss is calculated by categorical cross-entropy.

3.1.1 CIFAR-10 dataset

The CIFAR-10 dataset consists of 50,000 training images and 10,000 testing images of 10 balanced classes, each image is in color format (i.e. with 3 channels) and the size is 32x32 pixels. The 10 classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck (labeled using 10 integers: 0-9). Example images and more details can be found on the CIFAR-10 official website (http://www.cs.toronto.edu/~kriz/cifar.html).

3.1.2 Simulation Settings

One virtual central node and several virtual client nodes are involved in this simulation system, simulation can be done locally using one machine. We assume no data or nodes will be dropped out from or added into the system during the entire training process. Subsets of CIFAR-10's training part (4000 or 40000 samples, shortened as '4k dataset' and '40k dataset') are used for training while the whole evaluation part is used for testing (10000 samples). Client nodes are in equal size and samples from available classes within a node are balanced. By default, local epochs are set to 1.

When the system starts, the central node will initialize a model with randomized weights (this is done automatically by the TensorFlow). We apply Federated Averaging in this system, weighted by the size of each node. Each global round consists of the following basic actions:

1. Client nodes load weight from the central node,
2. Client nodes train the model over their local data and save the history,
3. Client nodes return weights to the central nodes,
4. The central node aggregates the weights and evaluates the model.

After the simulation, we collect the evaluation results from the central node as the overall testing history and average the client nodes' training history as the overall training history.

By default, tests for each scenario are implemented using the 4k dataset under 2 class, 5 class and IID circumstances or using the 40k dataset under 2 class, 3 class, 5 class and IID circumstances (in the following text, if there's no extra explanation, we use this default setting when mentioning '4k dataset' and '40k dataset').

In most tests, we run the system for over 1000 global rounds, except for in pilot tests (run for 500 epochs) or in the part of sharing data (run for 200 epochs due to the high computational costs) or several tests using the 4k dataset (early-stopping of 500 rounds has been applied to tests using the 4k dataset).

3.1.3 Pilot Tests

Each pilot test is performed for 500 global epochs, and the tests are implemented in two aspects. From the first aspect, we perform the test under IID circumstances and use datasets of different sizes (i.e. 500, 600, 700, 800, 900, 1000, 1500, 2000, 3000, 4000, 8000, 40000).

From the second aspect, we perform the test with the 4k dataset under various non-IID or IID circumstances (i.e. 1-9 class non-IID, IID), since many previous works show that a non-IID training set can result in performance degradation in federated learning (Kairouz *et al.* 2019). We present the skewness in label distribution simply by controlling available classes in each node (there is an example in the Appendix).

Based on pilot tests' results we decide to implement the following simulations with both 4k and 40k dataset, under 2 class, 5 class and IID circumstances. Later when we compare the results using 4k and 40k datasets an inverted trend under 2 class circumstances is found. So that we add 3 class circumstances to testing with 40k dataset.

3.1.4 Poisoning

We test this scenario where there is an anomaly node that returns arbitrary weights at each global round. When testing model poisoning, this is achieved by initializing a model in TensorFlow at each round and return the automatically initialized weights. In the additional tests of data poisoning, this is achieved by training the anomaly node with the dataset in which all labels are shuffled. During the training process, we split the training set to 10 nodes as usual and added in an anomaly node.

Tests for model poisoning are performed with both 4k and 40k dataset. The anomaly node claims that its size is as large as a certain proportion of the total training set. This proportion is 1% or 10% when using the 4k dataset, and is 1%, 3% or 10% when using the 40k dataset. Additional tests of poisoning frequency (i.e. anomaly nodes upload weights by every n epoch) and poisoning data are performed with the 40k dataset under IID condition.

3.1.5 Client Size

In some tests, we distribute our data to 10 clients, in other tests to 40 clients. This raises concerns about the effect caused by the client size. In this scenario, we compare the training

performance between splitting the same amount of data to 10 clients and 40 client cases. Testing is performed with both 4k and 40k dataset.

3.1.6 Delayed Update

In this scenario, some nodes update at a slower speed than others. We split the training set to 40 nodes and test the situation from 2 aspects. Testing is performed with both 4k and 40k dataset.

From the first aspect, we test various proportion of delayed nodes. 20%, 50% and 80% (with 4k dataset) or 25%, 50% and 75% (with 40k dataset) of nodes are delayed, they load the weights at global round n, and upload the weights at global round n+3. New weights won't be loaded until current weights are uploaded. In later texts, we summarize the result to 'delay proportion' of '0', '<50', '50' and '>50'.

From the second aspect, we test various speeds of delayed nodes. 20% of the nodes are delayed (with both 4k and 40k dataset), they load the weights at global round n, and upload the weights at global round n+3, n+12 or n+30. New weights won't be loaded until current weights are uploaded. In later texts, we summarize the result to 'delay speed' in '0', '3', '12' and '30'. Larger value in delay speed means the delayed nodes act in slower speed (i.e. more serious postpone in uploading weights).

3.1.7 Sharing and Resampling Data

Logically we understand that 1 class non-IID never behaves better than random since locally there will be little effective update. And in pilot tests, other seriously non-IID datasets also shows a certain decrease in performance. Previous research has suggested a data-sharing strategy that can improve training performance when using 1 class non-IID data (Zhao *et al.* 2018). The author assumes a small IID set (its size is between 2.5% to 25% as large as total training dataset) can be published for warming up the model or sharing between nodes.

In this scenario, we attempt on 3 different methods of using the shared data. Since oversampling methods we used here are too costly, we tested only with 4k dataset under 1 class and 5 class non-IID circumstances for 200 rounds, with '2.5%' sharing amount.

Using this strategy, after receiving the shared data, each client node will contain a mixed sample pool of shared and local data. Method 1 is the direct usage of the entire pool. Method 2 will under-sample a balanced subset from the pool (using ' RandomUnderSampler' from the ' imblearn' package). Method 3 will oversample the pool (we test both ' SMOTENC' and ' ADASYN' from the ' imblearn' package).

3.2 Cloud Implementation

OIA-DDR dataset is used for cloud implementation. The task for the implementation is to input images of size 224x224x3 and classify them into 2 categories. VGG16 network (pre-

trained on ImageNet database) is used. We apply the ADAM optimizer with the learning rate initialized at 1e-4 and decayed by rate 0.05 in every epoch or round. The batch size is set to 300; the loss will be calculated by binary cross-entropy.

$$\text{learning\_rate} = 1e\text{-}4/(1+0.05*\text{epoch\_number})$$

3.2.1 OIA-DDR dataset

The OIA-DDR dataset provides high-quality diabetic retinopathy (DR) images and annotations. The grading annotation labels the images to six classes: no DR, mild, moderate, severe, proliferative and ungradable. In this project, we neglect images from the 'ungradable' class, using 6266 images from 'no DR' class as healthy samples and 6256 images from the rest 4 classes as DR samples.

Images in this dataset are in different sizes and qualities. We preprocess the images using the methods from Kaggle's public kernels (links in the Appendix), crop and resize them to 224x224 pixels.

In our implementation, we select 10000 balanced samples and divide them into a slightly non-IID training set (assume there are a large and balanced client and 2 small and unbalanced clients). The rest 2521 samples are used for testing. An IID version of the training is also tested (which is an ideal scenario).

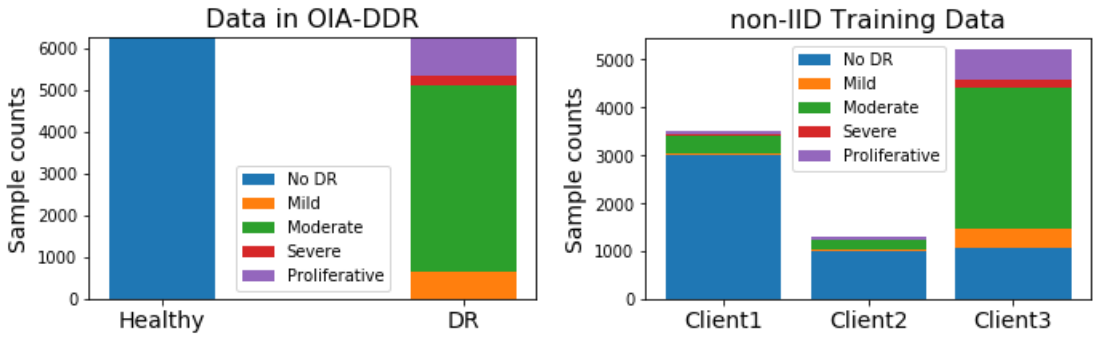


Figure 1. Sample distribution. *Left*: Samples in OIA-DDR dataset. *Right*: Sample distribution in the non-IID training set.

3.2.2 Cloud System

In the cloud implementation experiment, our synchronous gRPC system consists of 1 central server and 3 clients. The clients join in the training by sending a request to the server by every round. The server knows it will have 3 clients; aggregation will start when all 3 returned weights are received. Each global round consists of the following basic actions (from the server's side):

0. The server initializes the model and saves the weight,
1. The server receives requests,

2. Server register clients and send weights,
3. (Clients perform local training),
4. Server receive returned weights, wait until 3 weights are ready,
5. The server aggregates weights and performs an evaluation.

As we mentioned before, an IID and a slightly non-IID dataset are prepared for this task. We also test the centralized training performance with the total training test and the largest federated subset.

4 Results

4.1 Scenarios testing

In an ideal training process, the best accuracy and minimal loss will be achieved at the convergence epoch. In our training, we have observed a slight increase in accuracy after the loss reaches to its least, in which case the model starts to overfit the training data. Since our loss function is cross-entropy, this situation is possible, for example, when some samples are classified to more deviated predictions from correct results while slightly deviated predictions are corrected. In such cases, it is hard to tell if an increase in accuracy can represent the increase in performance. Also, other better models might be able to avoid this situation. Thus, we assume our model performs best at the global round with minimal loss, the accuracy at that round represents the model's best performance. In our following plots, we will label this point by a colored square.

Since the original results have fluctuations, we show and manipulate on windowed testing accuracy and loss in this report, with window size equals to 21. In this section, we show statistical summaries or part of our testing results. The full and original records are available in the GitHub repository, including the accuracy and loss curve for both training and testing datasets. Also, we will append the windowed results at the end of the report.

4.1.1 Pilot Tests

Figure 2 compares centralized and federated learning's behavior with various sizes of the training dataset. The results can fluctuate due to machine learning's randomness, but it is obvious that the performance of both federated and centralized learning is improved when the size of training sets become larger, and testings in federated settings always converge at a slower speed than the ones in centralized settings. When the federated dataset is IID, there's no obvious difference between federated and centralized learning in final accuracy.

Figure 3 shows federated learning at different non-IID levels. It is obvious that accuracy and convergence speed decreases when the degree of non-IID increases.

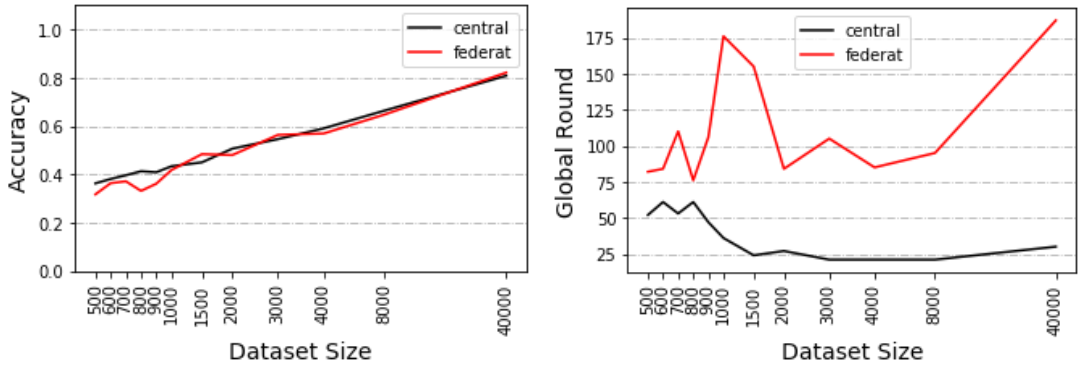


Figure 2. Pilot tests comparing various dataset size, performed under IID circumstance. *Left*: Accuracy at minimal loss. *Right*: Round index at the minimal loss.

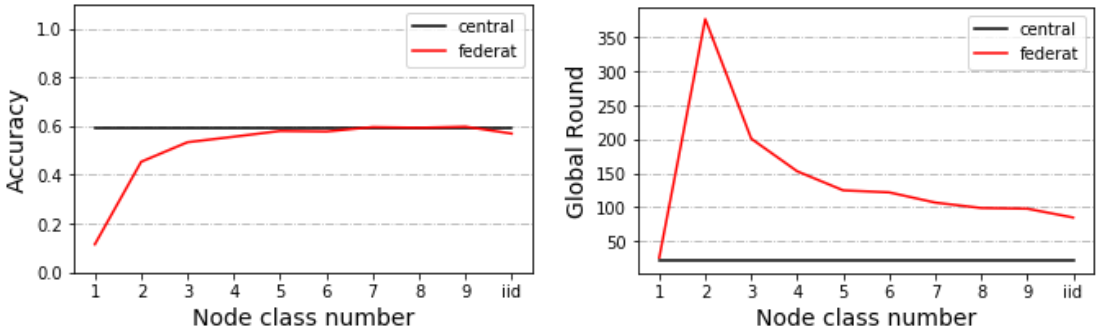


Figure 3. Pilot tests comparing various IID or non-IID circumstances, performed with 4k dataset. *Left*: Accuracy at minimal loss. *Right*: Round index at minimal loss.

We also have observed a trend when testing between dataset of size 4000 and 40000 under 2 class non-IID circumstances. In other tests, training with the larger dataset always results in higher accuracy, while in this test the trend is inverted. This inverted trend is also detectable in other 2 class non-IID circumstances, indicating a possibility that training with datasets of different sizes or skewness in distribution may result in different behaviors. Thus, we decide to test the following scenarios with both large (40k) and small (4k) datasets, and seriously non-IID (2 or 3 class), weakly non-IID (5 class) and IID datasets.

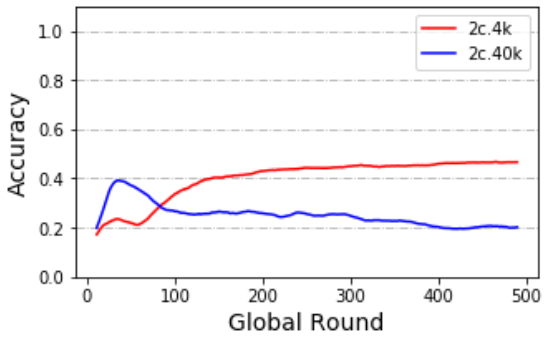


Figure 4. Test record of training with 4k and 40k dataset under 2 class non-IID circumstance.

4.1.2 Poisoning

The presence of an anomaly node leads to performance degradation. We use ‘p%-f’ to represent that the system is affected by the anomaly node of size p% at every f epochs. If using ’p%’, it is equal to ’p%-1’.

The influence is related to both the size of the anomaly node and the poisoning frequency. Similar accuracy trends among the situation ‘1%-1’, ‘3%-3’ and ‘10%-10’ have been observed, while there is a huge difference among ‘1%-1’, ‘3%-1’ and ‘10%-1’ or ‘10%-1’ and ‘10%-10’. In the 3 similar situations, they have the same average anomaly node size per round. The influence of anomaly node size, in this case, might still exist. Loss of ‘10%-10’ stops decreasing at earlier rounds than ’1%-1’.

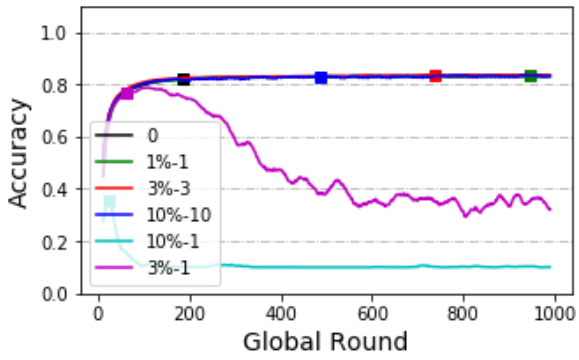


Figure 5. Accuracy curve of model poisoning tested with 40k dataset under IID circumstance.

Models trained with large dataset seems to have a higher tolerance to attacks from anomaly nodes. In the case of the 1% size anomaly node, we cannot observe performance degradation in model trained with an IID 40k dataset, while this degradation is obvious in models trained with 4k dataset. On the other hand, no obvious decrease pattern among non-IID groups has been observed. Though we have observed a slight decrease in some non-IID groups (while there is no decrease in the others), the decrease pattern is still unclear.

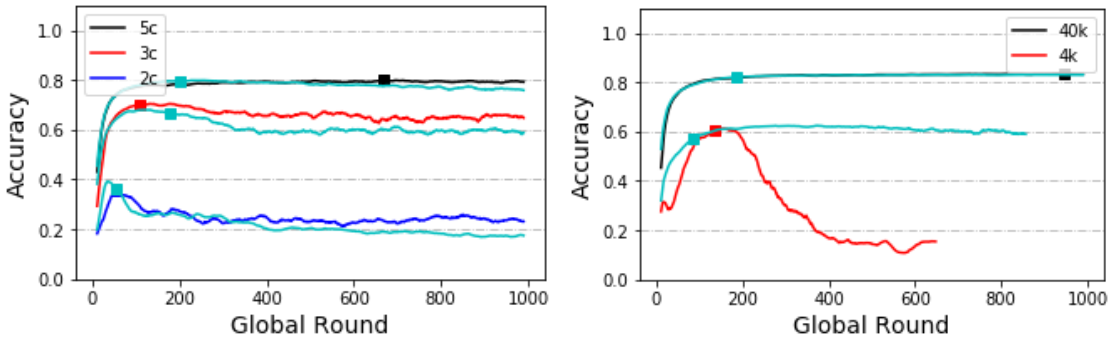


Figure 6. Accuracy curve of model poisoning. The light blue line without any label is the original curve without any poisoning for each case. *Left*: Testing with 40k dataset and ‘1%’ anomaly node. *Right*: Testing under IID circumstance and ‘1%’ anomaly node.

In some other cases, anomaly nodes may upload training results over the dataset with the shuffled labels, which is called data poisoning. Testing results in data poisoning indicates that data poisoning is weaker than model poisoning.

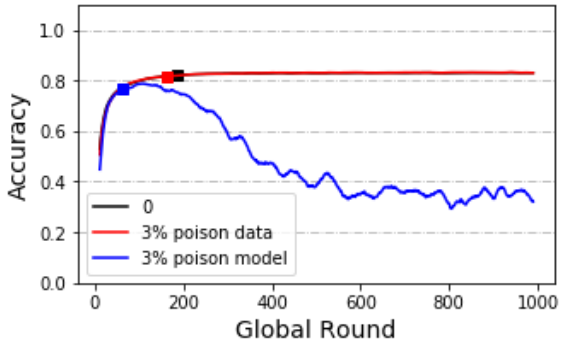


Figure 7. Accuracy curve of data poisoning and model poisoning. Testing with 40k dataset under IID condition and '3%' anomaly node.

4.1.3 Client Size

From Figure 8 below we can see training with 40 client nodes is slower than training with 10 client nodes while accuracy at minimal loss is not obviously affected. Statistical tests are applied and the results show a significant difference between convergence speeds of the two groups (p-value=0.0059). Differences in accuracy at minimal loss are not significant (p-value=0.71).

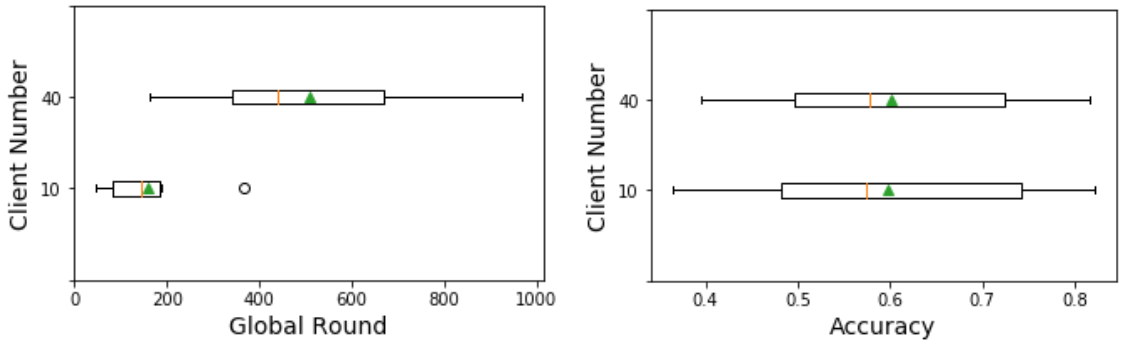


Figure 8. Box plot of the round index (*Left*) and accuracy (*Right*) where loss is the minimum, group by client number.

4.1.4 Delayed Update

Statistical tests are applied to detect the difference between no-delay-node cases and other situations. Statistical tests show a significance between with or without delayed nodes in the speed to reach minimal loss (round index) when delay speed is 3 and proportion between 20% to 25% (p-value=0.016). Surprisingly, this significance disappears as delay speed and proportion increases. On the other hand, when grouped by delay proportion, there is also no significance between group '<50' and '50' or '50' and '>50', making the trend of these two groups ambiguous.

When grouped by delay speed, a significance between group '3' and '12' has also been detected. Delayed nodes' effect on the speed to reach minimal loss (round index) is very likely to be weakened when delay speed becomes larger.

Also, accuracy at minimal loss with delay proportion larger than 50 is reported to be significantly reduced (p-value=0.01 and 0.005). Change in delay speed does not obviously affect the accuracy at minimal loss.

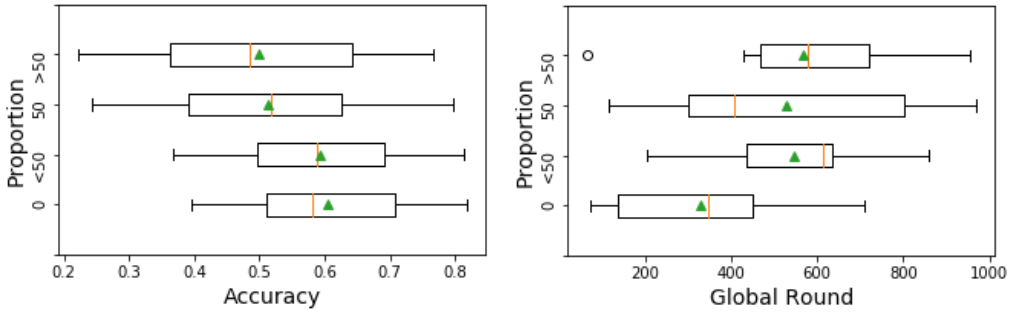


Figure 9. Box plot of the round index (*Left*) and accuracy (*Right*) where loss is the minimum, group by the proportion range of delayed nodes when delay speed set to 3.

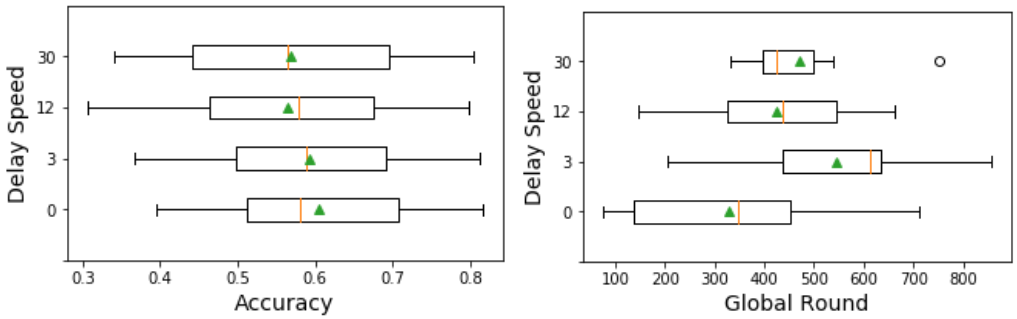


Figure 10. Box plot of the round index (*Left*) and accuracy (*Right*) where loss is minimum, group by delay speed, when delay proportion ranges in '<50'.

4.1.5 Sharing and Resampling Data

In this scenario, we compare different methods of using shared data. As we mentioned before, each client owns a mixed pool of local and shared data. When the shared amount is small (in this case, 2.5%), data in the pool is still unbalanced.

Figure 11 shows the results between directly using the mixed dataset and doing resample on the mixed dataset when sharing data is as large as 2.5% of the total dataset, tested with 4k dataset under 1 class and 5 class non-IID. Oversampling strategies seem to converge at a faster speed than the share-only method and under-sampling method. When data is seriously non-IID (the 1 class cases), all those methods perform better than without sharing but still cannot reach the accuracy level of the IID case. While the data is not seriously non-IID (the 5 class cases), share-only and oversampling methods only have a subtle influence on the

accuracy curve, while the under-sampling method seems to have placed a negative influence on original accuracy.

Also, the oversampling methods we tested here require a large amount of computing. Considering the fact that the share-only method can be much convenient and have similar or even better results than the resampling ones, in our case, this method seems to be the best strategy of using shared data.

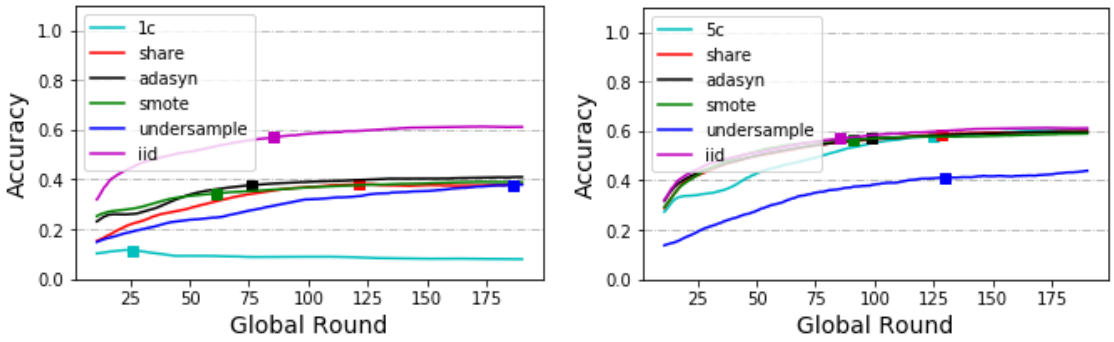


Figure 11. Accuracy curve of various sharing and resampling strategies, testing with 4k training set and 2.5 % shared set. *Left*: Results for 1 class non-IID circumstance. *Right*: Results for 5 class non-IID circumstances.

4.2 Cloud implementation

We have successfully performed our example task on the cloud. Similar as in simulation with CIFAR10, slightly non-IID and IID federated set behaves similarly. They perform worse than centralized training but better than training with the largest subset. Note that in Figure 12 we zoomed the Y-axis to show a much clearer difference, the results in the original scale are available in the Appendix.

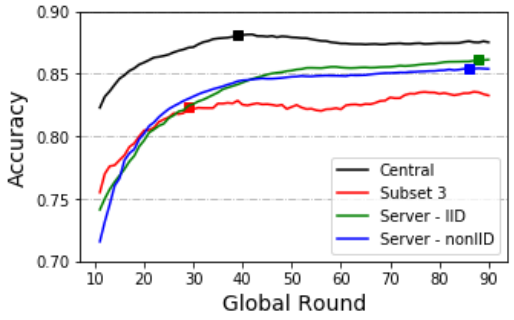


Figure 12. Accuracy curve of implementations with the OIA-DDR dataset.

5 Discussion

From previous researches, we already know that model or data poisoning can result in performance reduction and many defense methods have been suggested for this issue

(Kairouz *et al.* 2019). But we have not found a test to compare this effect in different training scenarios. In our testing, we have observed that model poisoned by the same average strength per round ('1%-1', '3%-3' and '10%-10') have similar behavior. We have also observed a stronger resistance to model poisoning when training with a larger dataset and we show a weaker effect of data poisoning. These results may indicate that when defense methods are not working for our system and checking at every round is too costly, we can reduce the checking frequency for very small or slow nodes since the system is comparatively less affected in slight or low-frequency attacks.

In testing with the delayed update, we have observed the system acting slower when the delay happened. This decrease in speed seems to be reduced when delay speed increases. A possible explanation of this is that delayed nodes of larger speed upload their weights at a lower frequency than others. Due to the ambiguous statistical results, we are still uncertain about the larger delay proportion's effect on speed to reach minimal loss. The situation of '>50' can be viewed as some small proportion of nodes acting at a faster speed than the others, while the rest of the system is 3 times slower than the original one. Logically, the overall system should act slower, but we cannot make a conclusion from our current results.

We have observed a decrease in accuracy when delay proportion is equal to or larger than 50%. The raise in delay speed does not obviously influence the final accuracy, possibly because the proportion of delayed node of this test is small. Considering its uploading frequency, we expect accuracy to be less affected when delay speed is increased since the weights from the delayed nodes can be viewed as slightly poisoned nodes. On the other hand, since data in the delayed part is not well trained, the system has less effective training data than usual, accuracy should be lower than usual. But these opinions need more testing. Our current results suggest that the majority of nodes should be synchronized when designing the federated system.

In the test with client size, we detect no decrease in accuracy, thereby increasing the confidence of inviting small clients to the system. However, we are only testing the cross-silo cases, where clients still hold a certain amount of local dataset and will not drop out. In cross-device cases, in the most extreme cases, there could be millions of clients and each client is a sample. In such cases, many other aspects need to be considered. Also, in our experiments, we assume all clients can afford computation resources for training. There could be the case, for example, where clients cannot provide surplus computing resources for training. In such cases, hiring cloud resources still face privacy issues. Techniques like pruning can save resources, but we are unsure about its effect in federated settings.

In the test with share data strategy, only when the training data is seriously non-IID, we found that the performance can be improved significantly by using shared data. We suggest using a share-only strategy in simple applications since it can be achieved in a much easier way and seems to have similar or better performance than the resampling methods. Current methods we used for oversampling fail to improve the share-only strategy, possibly because these

methods are not specific for generating images. ADASYN and SMOTE generate new images pixel by pixel, using the information from the original samples. This is not a suitable method for generating images, since objects in original images may have different colors, shapes, positions, and so on. In the future, if there are more suitable oversampling tools developed for images, such as using suitable augmentation, the performance of non-IID systems may be significantly improved, even without sharing data. However, our current result cannot support this view. For people who have not much experience in such fields, share-only is always the most convenient method.

Except for sharing the global data, as mentioned in the background section, there are other methods to solve data heterogeneity. FedMA (Wang *et al.* 2020) matches and averages the hidden elements. This method seems to work for a wide range of machine learning models, including CNNs and LSTMs. FedProx (Li T *et al.* 2018) adds a restriction factor between local and global weights. These methods can work better than FedAvg when facing non-IID datasets (FedProx can also reduce the effect of data or model poisoning), but they are still worse than the centralized implementation.

We also have thought about shifting evaluation methods to the clients. Ideally, if the clients can split part of local data as the testing set, aggregated evaluation results will be the same as collecting this testing set to the central. But in this case, we cannot evaluate the honesty of the clients. Thus, a valid centralized testing set is still a necessity for federated learning.

We successfully implemented a federated procedure on the cloud and get similar trends as in simulation tests. However, in this case, our weight size is around 80.6 Mb and the system contains only 3 clients. There might be some limitations in transporting size, though usually, deep learning models will not be that large. Another student in the Lab is focusing on communication research.

Also, the training design for DR detection seems to be less stable than the design of CIFAR-10 classification. When designing the model and training hyperparameters for DR detection, since the DR dataset is much more complex than the CIFAR-10 dataset, we use a larger network with pre-trained weights (trained on ImageNet database) and a relatively small learning rate, so that we can save the training efforts over basic-level features (i.e. angles, borders). Compared to CIFAR-10 classification, DR detection is more sensitive to parameter changes. Slight changes in learning rate or not using pre-trained weights can case a failer in converging. So we can expect this DR system to be weaker in resisting poisoning or delayed nodes (since the size of the DR dataset is too small for processing complex images), but further research is still needed.

6 Conclusion

We have tested several scenarios in federated learning under various circumstances, the results can provide an easy-to-understand reference to future federated designs. Our simulation system can also be edited for other scenario testings with tiny changes.

We also perform a trivial example of diabetic retinopathy detection on the cloud. Currently, we have not found many federated implementations in biological and medical fields. Our code can be an easy-to-understand example for bioinformaticians or other people who would like to attempt on federated machine learning.

7 Acknowledgment

I want to acknowledge suggestions and helps from my supervisor and subject reader and members in the lab. And Mattias Åkesson who provide the model and instructions for simulating federated learning.

8 References

Angermueller C, P?Rnamaa T, Parts L, Stegle O. 2016. Deep learning for computational biology. Molecular Systems Biology 12: 878.

Benitez CMV, Chidambaram C, Hembecker F, Lopes HS. 2011. A Comparative Study of Machine Learning and Evolutionary Computation Approaches for Protein Secondary Structure Classification. InTech

Chen X, Ishwaran H. 2012. Random forests for genomic data analysis. Genomics 99:

Cubuk ED, Zoph B, Mane D, Vasudevan VK, Le QV. 2018. AutoAugment: Learning Augmentation Policies from Data. arXiv: Computer Vision and Pattern Recognition

Eraslan G, Simon LM, Mircea M, Mueller NS, Theis FJ. 2019. Single-cell RNA-seq denoising using a deep count autoencoder. Nature Communications 10: 390.

He Chaoyang, M. Annavaram, Avestimehr S. 2020. FedNAS: Federated Deep Learning via Neural Architecture Search. CVPR 2020 workshop

Jeong E, Oh S, Kim H, Park J, Bennis M, Kim S. 2018. Communication-Efficient On-Device Machine Learning: Federated Distillation and Augmentation under Non-IID Private Data. arXiv: Learning

Jurtz VI, Johansen AR, Nielsen M, Almagro Armenteros JJ, Nielsen H, Sønderby CK, Winther O, Sønderby SK. 2017. An introduction to deep learning on biological sequence data: examples and solutions. Bioinformatics 33: 3685–3690.

Kairouz P, Mcmahan HB, Avent B, Bellet A, Bennis M, Bhagoji AN, Bonawitz K, Charles Z, Cormode G, Cummings R, others. 2019. Advances and Open Problems in Federated Learning. arXiv: Learning

Karimireddy SP, Kale S, Mohri M, Reddi SJ, Stich SU, Suresh AT. 2019. SCAFFOLD: Stochastic Controlled Averaging for On-Device Federated Learning. arXiv: Learning

Krizhevsky A. 2012. Learning Multiple Layers of Features from Tiny Images. University of Toronto

Li T, Gao Y, Wang K, Guo S, Liu H, Kang H. 2019a. Diagnostic Assessment of Deep Learning Algorithms for Diabetic Retinopathy Screening. Information Sciences 501: 511–522.

Li T, Sahu AK, Talwalkar A, Smith V. 2019b. Federated Learning: Challenges, Methods, and Future Directions. arXiv: Learning

Li T, Sahu AK, Zaheer M, Sanjabi M, Talwalkar A, Smith V. 2018. Federated Optimization for Heterogeneous Networks. arXiv: Learning

Li Y, Han R, Bi C, Li M, Wang S, Gao X. 2018. DeepSimulator: a deep simulator for Nanopore sequencing. Bioinformatics 34: 2899–2908.

Liu Y, Chen T, Yang Q. 2018. Secure Federated Transfer Learning. arXiv: Learning

Mcmahan HB, Moore E, Ramage D, Hampson S, Arcas BAY. 2016. Communication-Efficient Learning of Deep Networks from Decentralized Data. arXiv: Learning

Passeratpalmbach J, Farnan T, Miller RC, Gross MS, Flannery HL, Gleim B. 2019. A blockchain-orchestrated Federated Learning architecture for healthcare consortia. arXiv: Computers and Society

Poplin R, Chang P-C, Alexander D, Schwartz S, Colthurst T, Ku A, Newburger D, Dijamco J, Nguyen N, Afshar PT, Gross SS, Dorfman L, McLean CY, DePristo MA. 2018. A universal SNP and small-indel variant caller using deep neural networks. Nature Biotechnology 36: 983–987.

Rajpurkar P, Irvin J, Ball RL, Zhu K, Yang B, Mehta H, Duan T, Ding D, Bagul A, Langlotz CP, Patel BN, Yeom KW, Shpanskaya K, Blankenberg FG, Seekins J, Amrhein TJ, Mong DA, Halabi SS, Zucker EJ, Ng AY, Lungren MP. 2018. Deep learning for chest radiograph diagnosis: A retrospective comparison of the CheXNeXt algorithm to practicing radiologists. PLOS Medicine 15: 1–17.

Ramachandran P, Zoph B, Le QV. 2017. Swish: a Self-Gated Activation Function. arXiv: Neural and Evolutionary Computing

Real E, Liang C, So D, Le QV. 2020. AutoML-Zero: Evolving Machine Learning Algorithms From Scratch. arXiv: Learning

Sheller MJ, Reina GA, Edwards B, Martin J, Bakas S. 2018. Multi-Institutional Deep Learning Modeling Without Sharing Patient Data: A Feasibility Study on Brain Tumor Segmentation. arXiv: Learning

Stokes JM, Yang K, Swanson K, Jin W, Cubillos-Ruiz A, Donghia NM, MacNair CR, French S, Carfrae LA, Bloom-Ackermann Z, Tran VM, Chiappino-Pepe A, Badran AH, Andrews IW, Chory EJ, Church GM, Brown ED, Jaakkola TS, Barzilay R, Collins JJ. 2020. A Deep Learning Approach to Antibiotic Discovery. Cell 180: 688-702.e13.

Wang H, Yurochkin M, Sun Y, Papailiopoulos DS, Khazaeni Y. 2020. Federated Learning with Matched Averaging.

Xu M, Zhao Y, Bian K, Huang G, Mei Q, Liu X. 2020. Federated Neural Architecture Search. arXiv

Yao Q, Wang M, Escalante HJ, Guyon I, Hu Y, Li Y, Tu W, Yang Q, Yu Y. 2018. Taking Human out of Learning Applications: A Survey on Automated Machine Learning. arXiv: Artificial Intelligence

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

9 Example Non-IID Distribution

2 class non-IID, 4k dataset (CIFAR-10), 10 worker

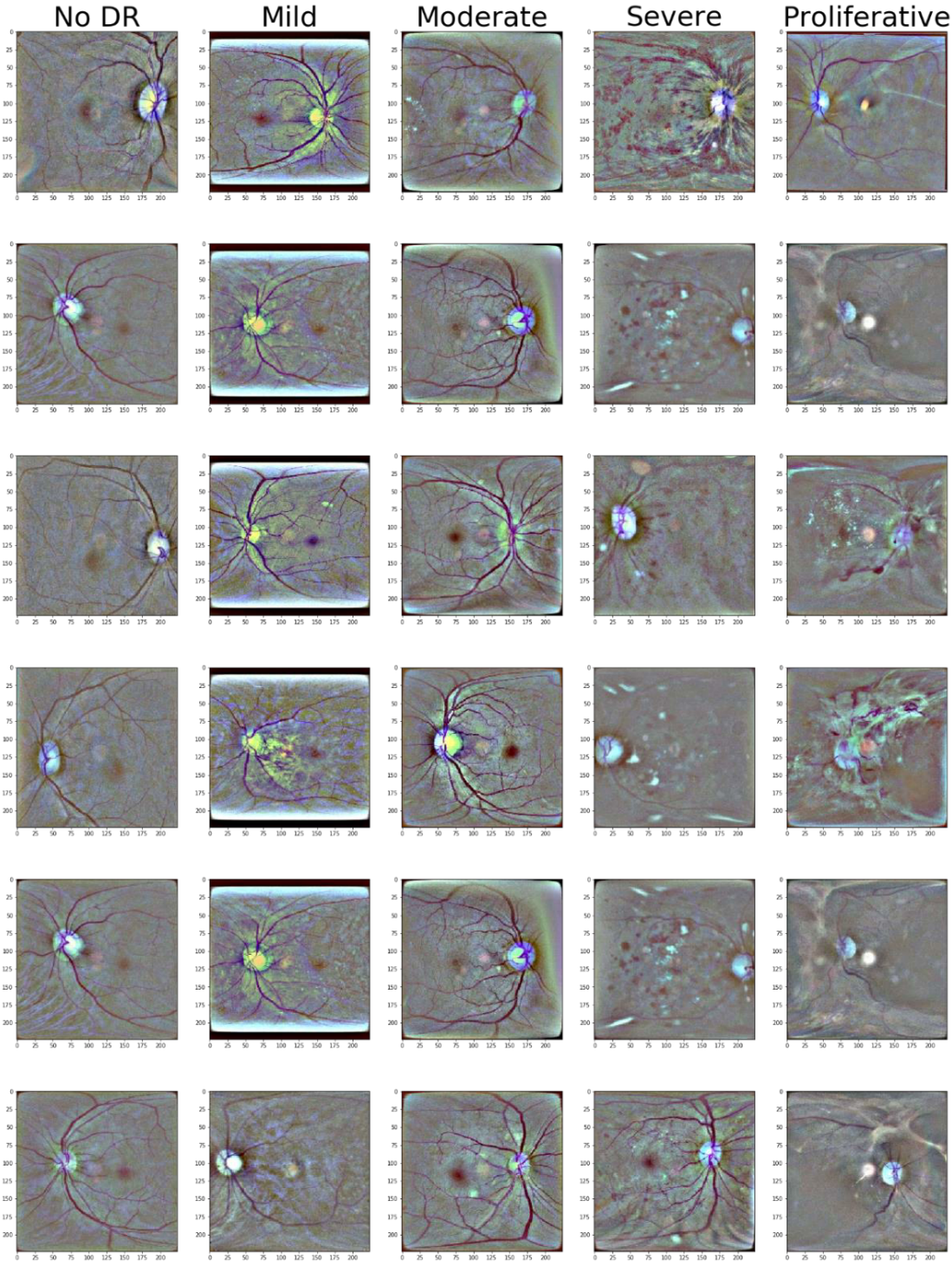
Node 1	Classe 1 * 200 + Class 2 *200
Node 2	Classe 2 * 200 + Class 3 *200
Node 3	Classe 3 * 200 + Class 4 *200
Node 4	Classe 4 * 200 + Class 5 *200
Node 5	Classe 5 * 200 + Class 6 *200
Node 6	Classe 6 * 200 + Class 7 *200
Node 7	Classe 7 * 200 + Class 8 *200
Node 8	Classe 8 * 200 + Class 9 *200
Node 9	Classe 9 * 200 + Class 10 *200
Node 10	Classe 1 * 200 + Class 10 *200

*No overlap in samples between nodes.

*Each client is a node.

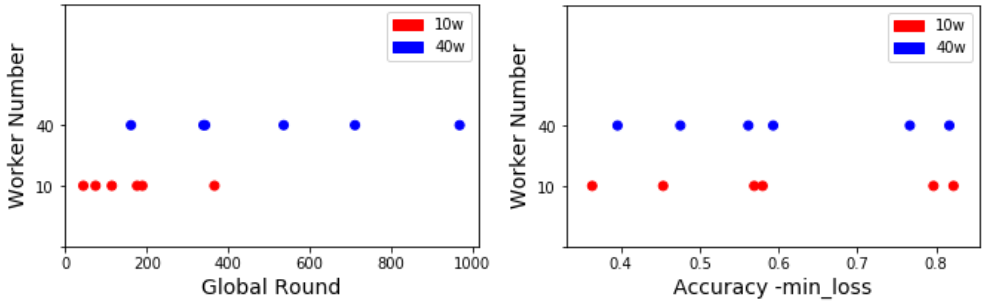
10 Appendix - OIA-DDR dataset after pre-processing

Methods from (https://www.kaggle.com/ratthachat/aptos-eye-preprocessing-in-diabetic-retinopathy) and (https://www.kaggle.com/titericz/circle-to-rectagle-preprocessing-1) are used for preprocessing original images to our input data.

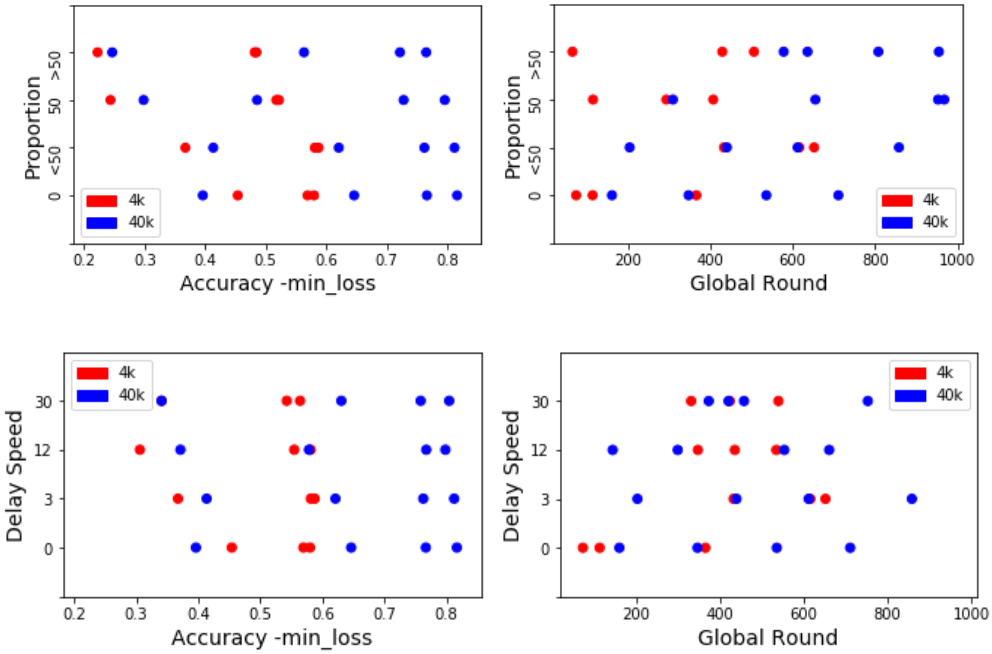


11 Appendix – Samples used in Box plots

Client Size



Delayed Update



12 Appendix - Statistical Records

In this Appendix, we record the results of statistical tests. Shapiro–Wilk test and Levene's test have been performed to assess normality and variance homogeneity, which is the pre-requirement of using the T-test. We perform the T-test or Wilcoxon test depending on assessing results. Results tested by the Wilcoxon test are tagged with '#'.

Client Size

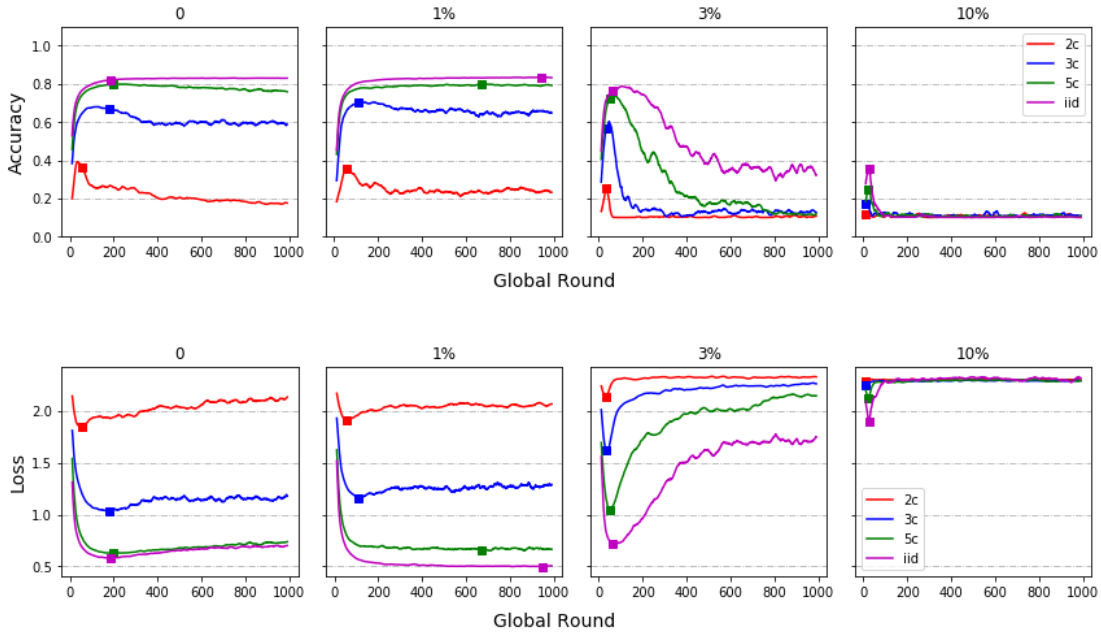
	'10W'-'40W'
ACC	0.71
ROUND	0.0059 *

Delayed Update

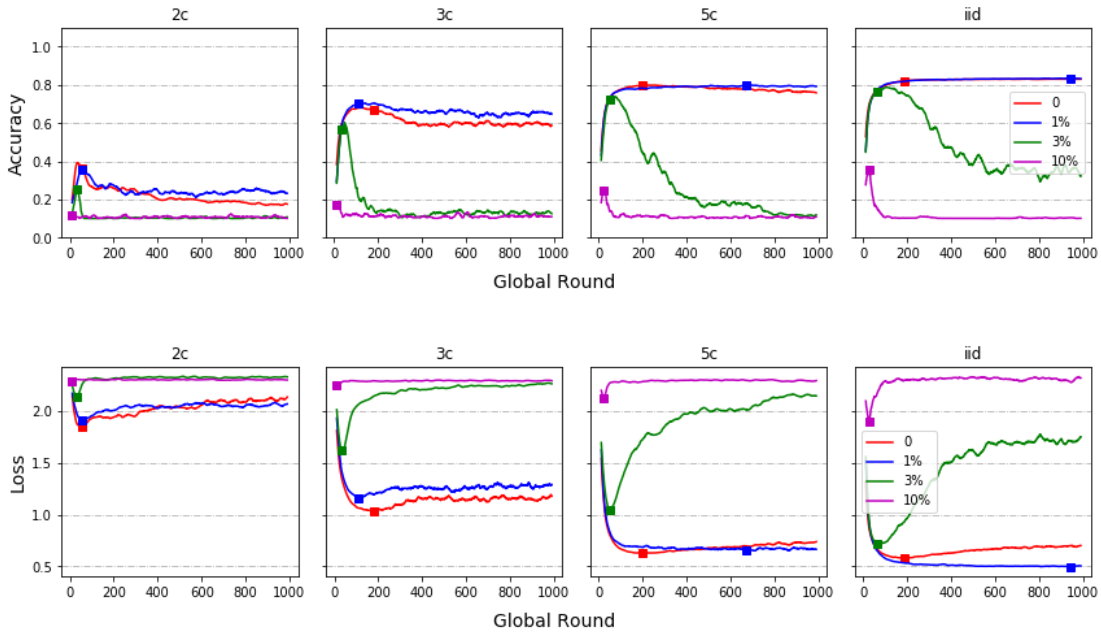
PROPORTION	**'0'-'<50'**	**'<50'-'50'**	**'50'-'>50'**	**'0'-'50'**	**'0'-'>50'**	**ANOVA**
ACC	0.41	0.0041 *	0.39	0.01 *	0.005 *	0.62
ROUND	0.016 *	0.90	0.60	0.089	0.092	0.35
DELAY SPEED	**'0'-'3'**	**'3'-'12'**	**'12'-'30'**	**'0'-'12'**	**'0'-'30'**	**ANOVA**
ACCURACY	0.41	0.027 *	0.77	0.11	0.052	0.97
ROUND	0.016 *	0.014 *	0.44	0.17	0.059	0.23

13 Appendix - Scenario testing results with 40k set

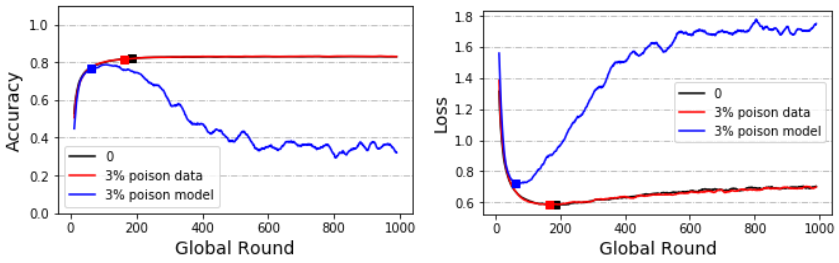
Model Poisoning – By Anomaly Node Size



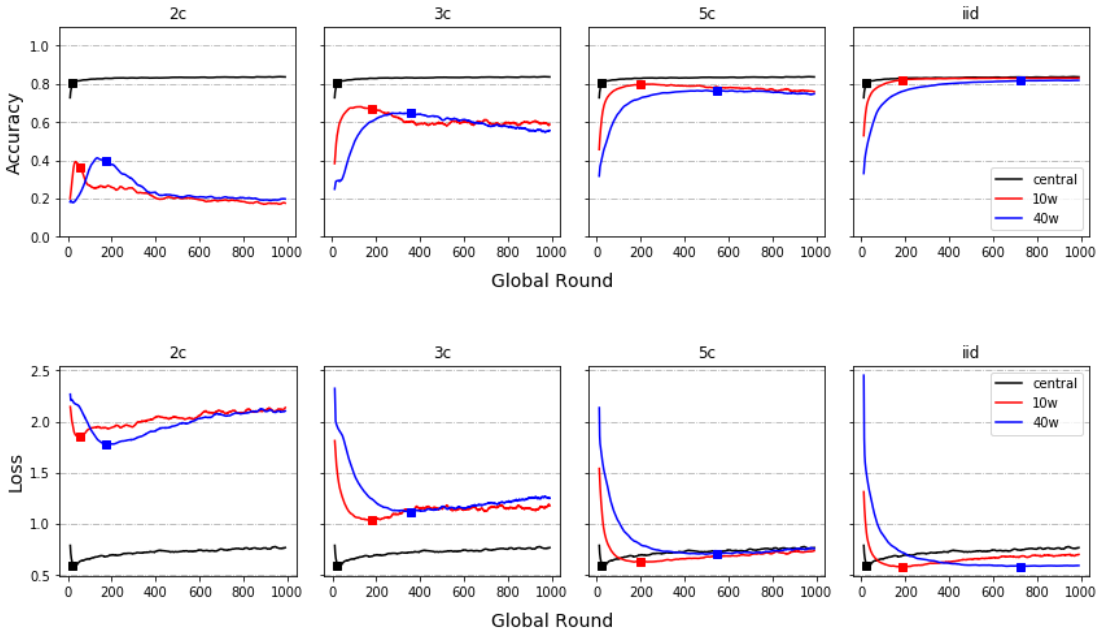
Model Poisoning – By IID/non-IID



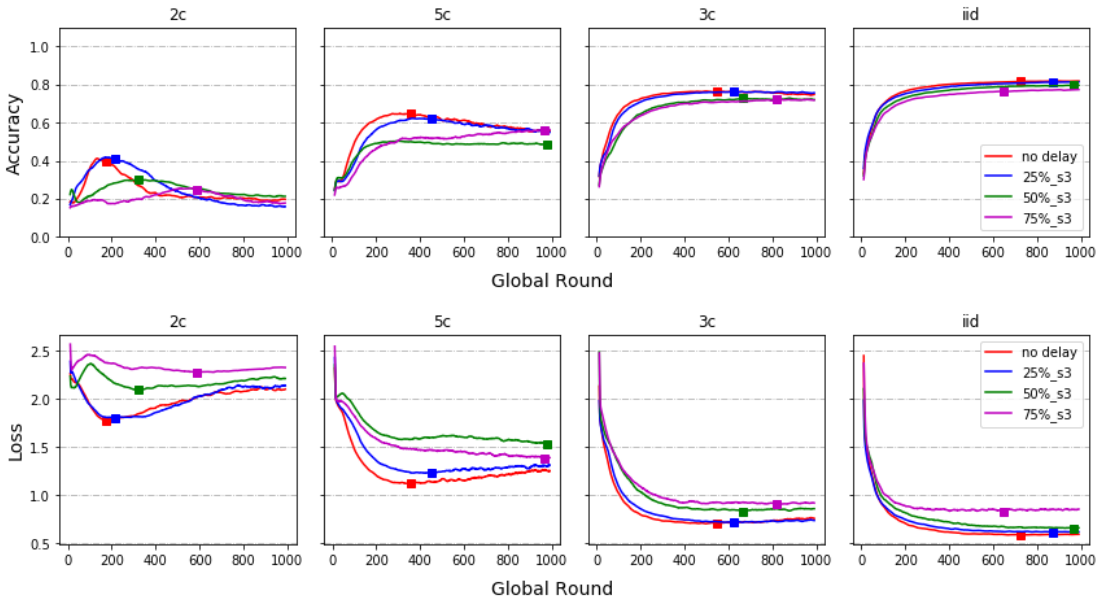
Data Poisoning



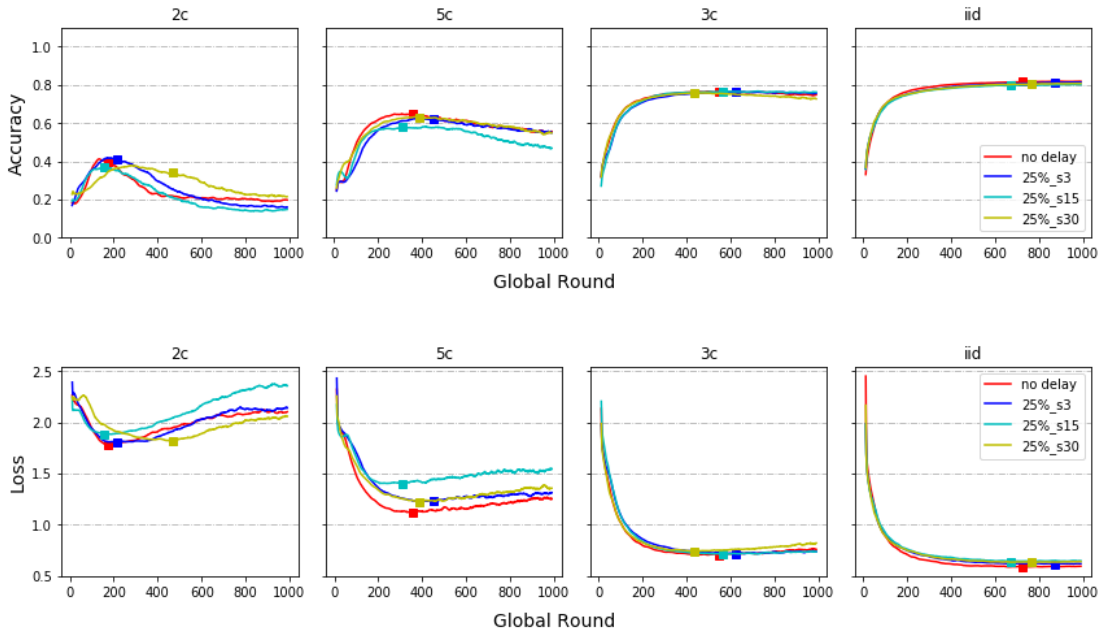
Client Size



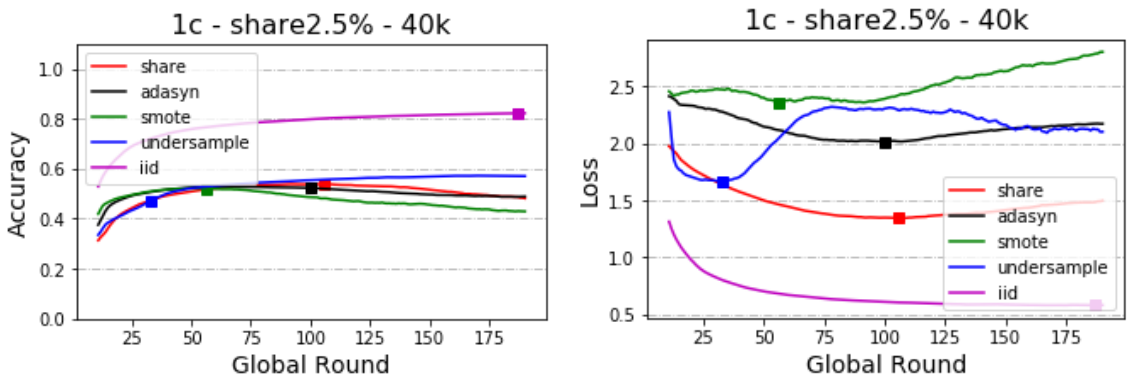
Delayed Update – By Proportion



Delayed Update – By Speed

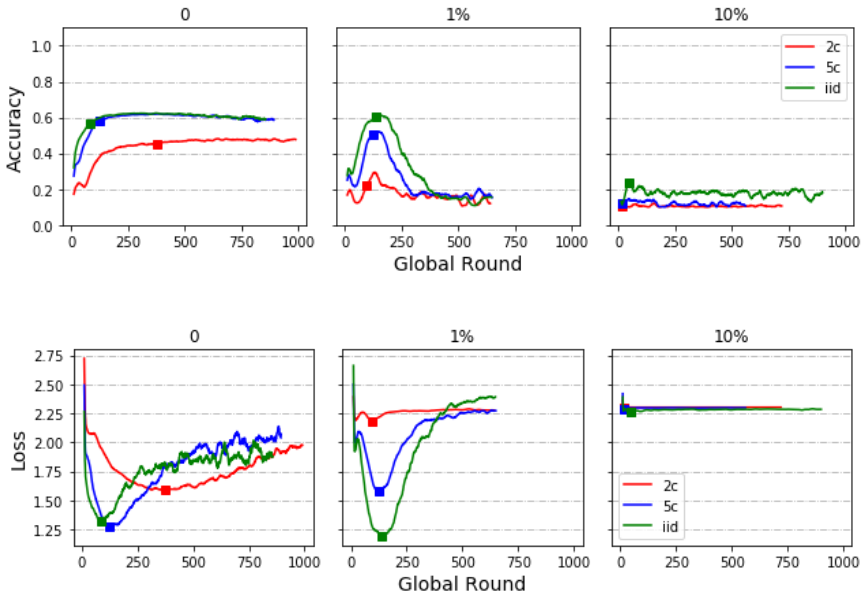


Share data strategy (Additional)

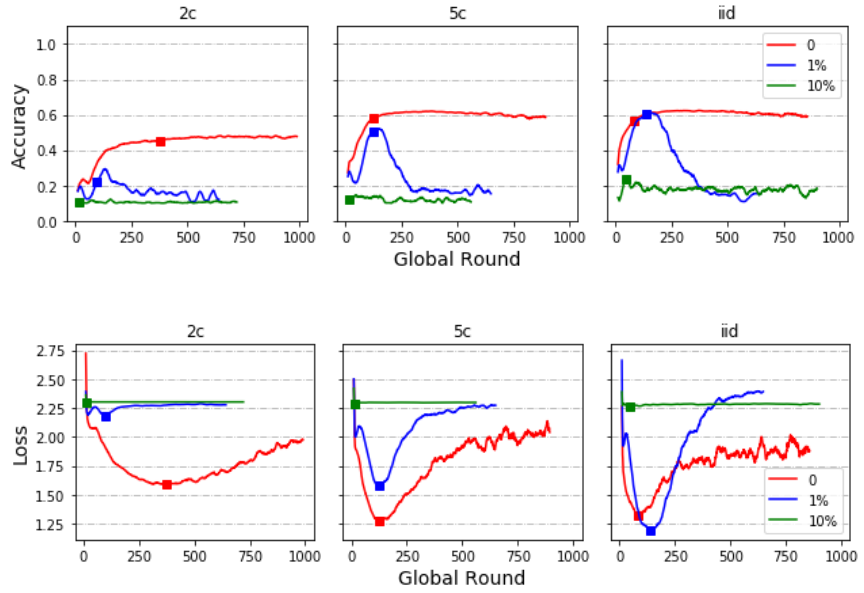


14 Appendix - Scenario testing results with 4k set

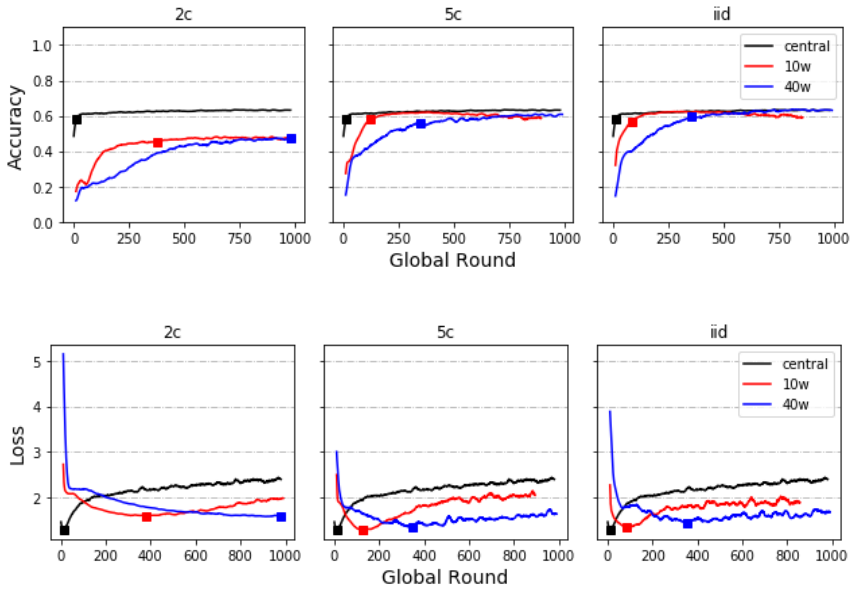
Model Poisoning – By Anomaly Node Size



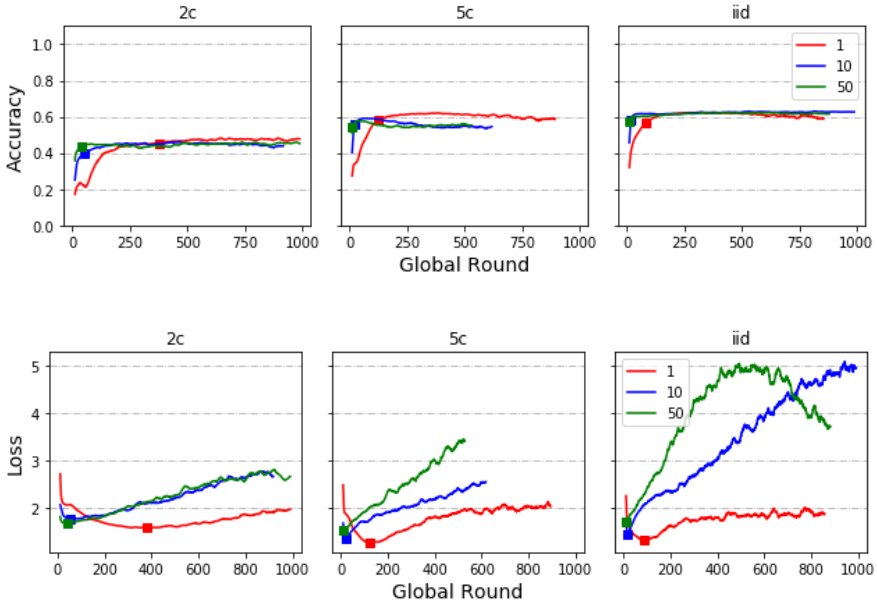
Model Poisoning – By IID/non-IID



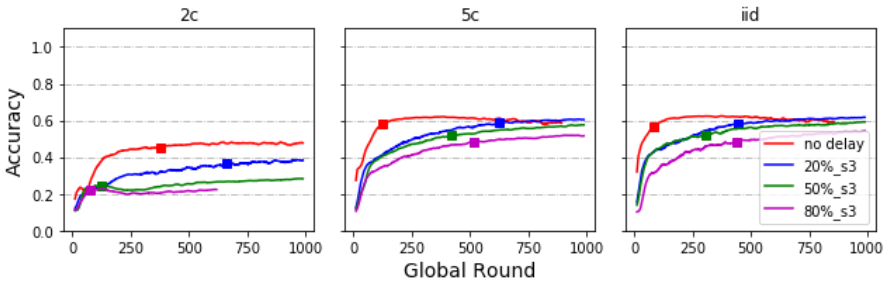
Client Size

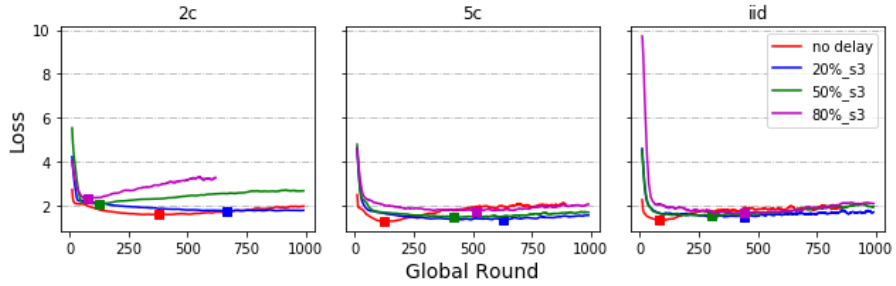


Local Round (Additional)

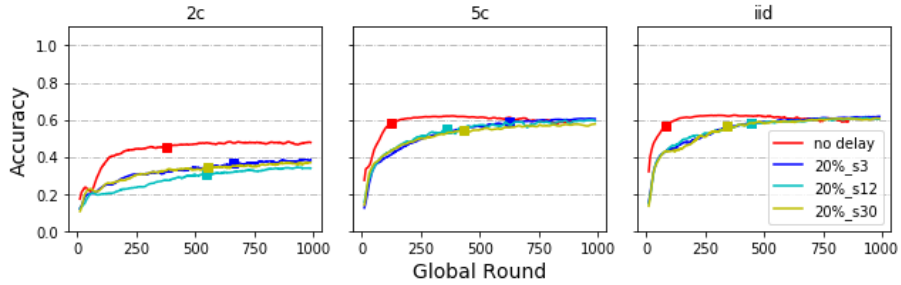


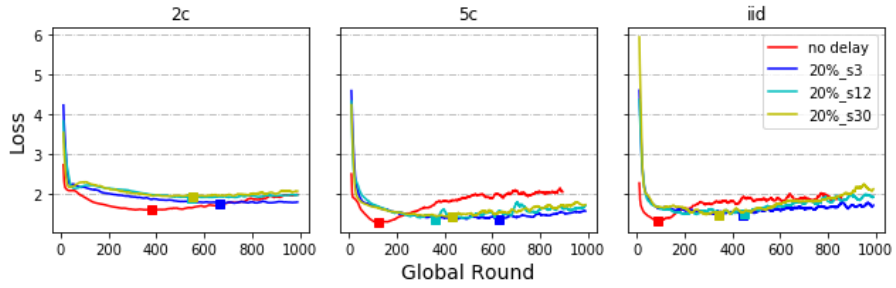
Delayed Update – By Proportion



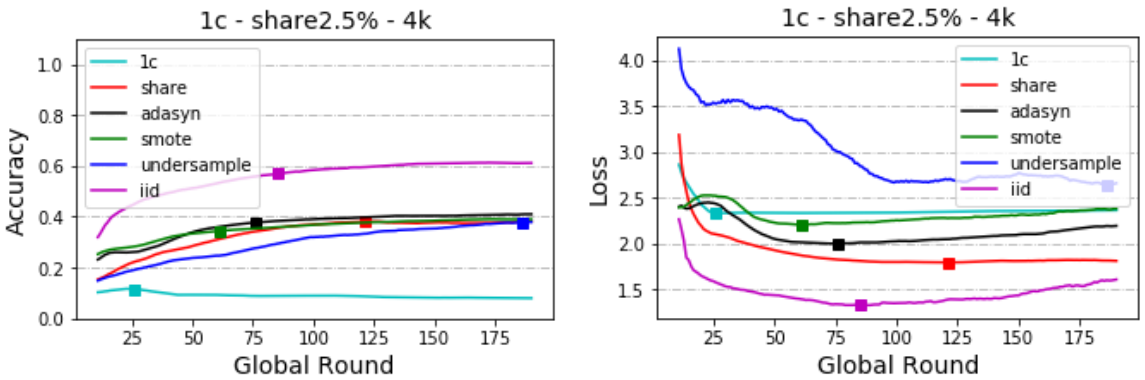


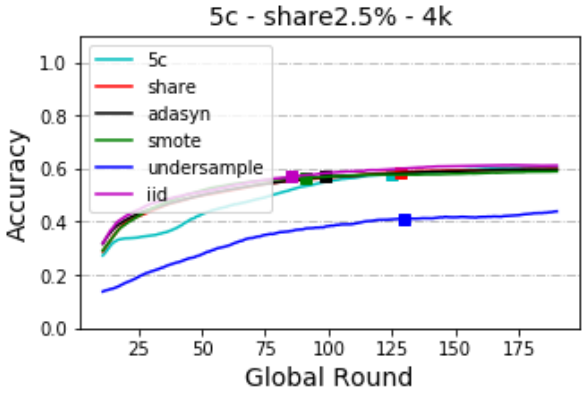
Delayed Update – By Speed

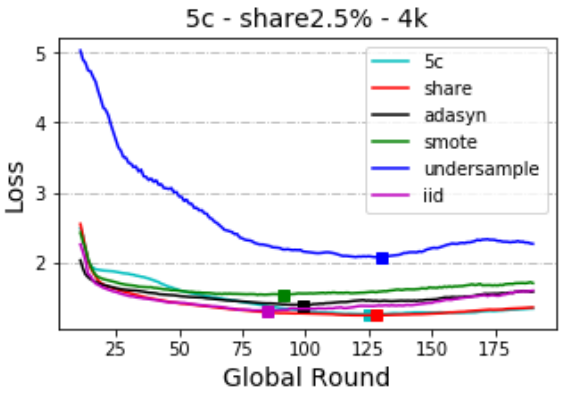




Share data strategy




5c - share2.5% - 4k
Accuracy
Global Round
5c
share
adasyn
smote
undersample
iid
0.0
0.2
0.4
0.6
0.8
1.0
25
50
75
100
125
150
175


5c - share2.5% - 4k
Loss
Global Round
5c
share
adasyn
smote
undersample
iid
2
3
4
5
25
50
75
100
125
150
175

15 Appendix - Results for cloud implementation