Abstract

Machine learning, especially deep learning, is becoming popular in biological and medical fields. However, researches using this technique are somehow prohibited by a lack of large and suitable dataset. A new machine learning setting called Federated learning is considered to have an ability of utilizing small and private datasets without leakage in privacy, giving more potential opportunities for future researches.

In this project, we test federated learning’s behaviors under various scenarios using a simple CIFAR10 dataset. We also provide an example federated diabetic retinopathy detection task on the cloud. We hope this work can be a reference for bioinformaticians who want to attempt on federated settings in their researches.

Learning without sharing

Popular Science Summary

Jiarong Liang

Current day biological or medical researchers have produced a large amount of data, which seems to be a perfect feed to machine learning tasks. In fact, machine learning, especially deep learning is becoming more and more popular in such fields. In many cases, the researcher will ’teach’ a computer-based model for certain tasks using a large and suitable dataset. Many intreasting attemptions have already been done in developing drugs, assisting doctors, predicting epidemics and so on.

However, in most other cases, researches are prohibited by lacking suitable datasets, since biological or medical records are sometimes sensitive and not allowed to be shared with each other. For example, if researchers want to build an application for detecting a disease, he will need many patients’ medical record from the hospital which is obviously the patients’ privacy. In another example, some companies run a business for disease prodiction using individual’s gene sequence and thus own the records of their client’s genetic information (also is privacy). In other situations where privacy information can be deducted from the records, publictising can also be dangerous and illegal.

We all know that the more data you feed into the model, the better it will learned. Unfortunally, a suitable amount of data is not always available in single institution especiall for small instances. Since we cannot share these data, these small datasets become somehow wasted.

A new method has been developed for avoiding such limitation. This method, which is called Federated learning, is published in 2016 and is already been attempted in the development of many applications such as the Google keyboard. This method can make good use of client’s private datasets without having access to them. In this project, we will test this method’s performance under several possible scenarios and perform 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

ADAM	A Stochastic Optimization Algorithm
DR	Diabetic Retinopathy
FedAvg	Federated Averaging Algorithm
FedSGD	Federated Stochastic Gradient Descent Algorithm
HTTP/2	Hypertext Transfer Protocol Version 2
IID	Independent and Identically Distributed
RPC	Remote Procedure Call
TCP	Transmission Control Protocol
VGG16	A Convolution Neural Network Architecture

Software Links

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

Project GitHub Repository

https://github.com/Jiarong000/Thesis2020

1 Introduction

Machine learning methods has been applied to medical or biological researches for long. Traditional algorithms such as PCA, random forest, AdaBoost has been widely used (Benitez *et al.* 2011, Chen & Ishwaran 2012). In recent years, a sub field called deep learning has become 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. Beside these, many interesting works has been done in recent years. DeepVariant aims to detect genetic variants from sequencing data (Poplin *et al.* 2018). DeepSimulator attempts to generate 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, 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 will test federated learning's behavior under various scenarios and perform an example federated medical image classification task on cloud. We hope this work can provide refences for future federated designs and provide a trivial example for 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 will 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 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 central server and the most importantly, preserves each client's private data.

There are many domains of federated learning. Cross-device federated learning is implemented on 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 (Yang *et al.* 2019).

In most biological or medical use cases, cross-silo and horizontal federated learning will be performed. A success case of brain tumor segmentation in federated settings has been reported (Sheller *et al.* 2018).

2.2 Federated Algorithms

Federated Averaging (FedAvg) and Federated SGD (FedSGD) are basics for many optimization algorithms and are widely used. In FedAvg settings, client weights will be collected and averaged while in FedSGD settings, gradients will be collected. Many other optimization algorithms are developed for various purposes, (Kairouz *et al.* 2019) provides a summary of popular ones.

2.3 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. However, by the end of May 2020, their official versions have not yet been released, several simulation functions are available.

Another software called Federated AI Technology Enabler (FATE) has been released. FATE already have enabled several functions in both traditional learning and deep learning.

2.4 Communication tools

In 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.

3 Dataset and methods

In this project, the CIFAR-10 dataset (Krizhevsky 2012) has been used for locally simulating different scenarios of federated learning, the OIA-DDR dataset (Li T *et al.* 2019a) has been used for implementing an example medical image classification task on the cloud. The performance of the models will not be high enough for any actual usage, only for testing federated learning in this project.

Each simulation task has been tested for 1000 epochs (for centralized training) or global rounds (for federated learning) and the cloud implementation has been tested for 100 epochs/rounds. The code is written in Python3 and TensorFlow 2.0.1 is used.

3.1 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 and the size is 32x32 pixels.

3.2 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 public kernels on Kaggle, crop and resize them to 224x224 pixels.

In our implementation, we select 10000 balanced samples and divide it to slightly non-IID federate training set. The rest 2521 samples are used for testing. An IID version of training is also tested.

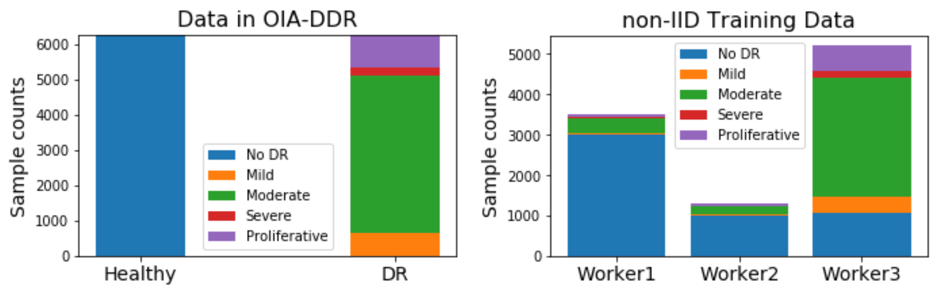


Figure 1. Sample distribution. Left: Samples in OIA-DDR dataset. Right: Sample distribution in non-IID training set.

3.3 Training designs

The task for CIFAR-10 is to input images of size 32x32x3 and classify them to 10 categories. The model is provided by Mattias Åkesson. We apply ADAM optimizer with learning rate set to 1e-3. The batch size is set between 50 to 100; loss will be calculated by categorical cross-entropy.

The task for OIA-DDR dataset is to input images of size 224x224x3 and classify them to 2 categories. VGG16 network (pre-trained on ImageNet dataset) is used. We apply ADAM optimizer with learning rate initialized at 1e-4 and decayed by rate 0.05 in every epoch or round. The batch size is set to 300; loss will be calculated by binary cross-entropy.

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

3.4 Scenarios testing

We simulate and test different scenarios in federated learning in this part. Due to the limit of time and computational resources, each test is performed only once.

3.4.1 Simulation System

One central node and several client nodes are involved in this simulation system. We assume no data or nodes will be dropped out from or added into the system during the entire training process.

When the system starts, the central node will initialize a model. We apply Federated Averaging in this system, weighted by size of the 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. Central node aggregates the weights and evaluates the model.

After simulation, we collect the evaluation results in central node's folder as the overall testing history and average the node's training history as the overall training history.

3.4.2 Pilot Tests

Before the formal implementation of scenario testing, we test the system with various sizes of dataset and various IID or non-IID classes. Figure 2 compares centralized and federated learning's behavior with various sizes of training dataset. The result may fluctuate due to randomness, but it is obvious that the performance of both federated and centralized learning can be improved with larger training set, and federated learning always acting at slower speed than the centralized learning.

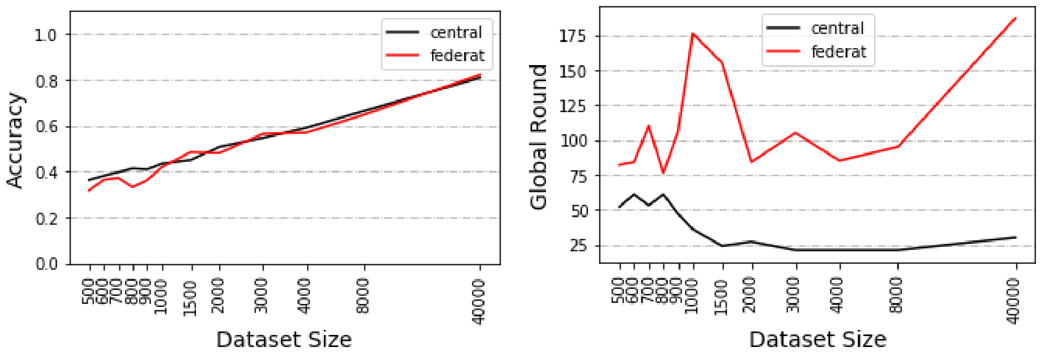


Figure 2. Test is performed under IID circumstance for 500 epochs and Manipulations are implemented on windowed accuracy and loss (window size equals to 21). Left: Accuracy at minimal loss under various dataset size. Right: Round index at minimal loss under various dataset size.

In previous works we already know that a non-IID training set can result in performance degradation in federated learning. There is no obvious standard to quantify the degree of non-IID, but researchers usually identify it by available classes in each node. Figure 3 shows federated learning's behavior at different non-IID levels. It is obvious that accuracy and speed will decrease when the degree of non-IID increases.

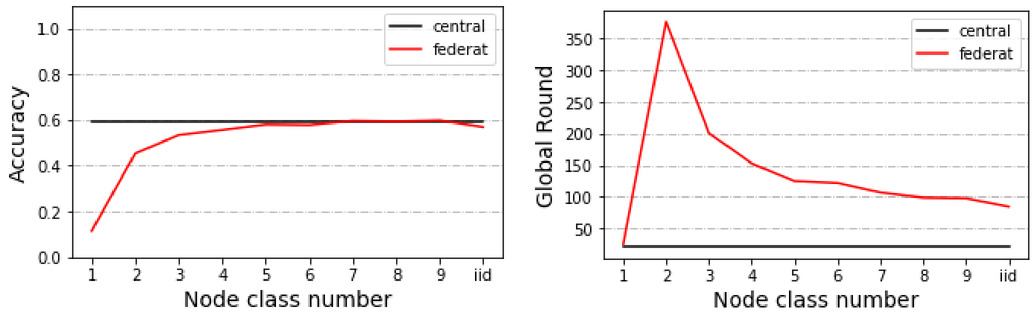


Figure 3. Test records with dataset of size 4000 for 500 epochs. Manipulations are implemented on windowed accuracy and loss (window size equals to 21). Left: Accuracy at minimal loss under various IID or non-IID circumstances. Right: Round index at minimal loss under various IID or non-IID circumstances.

We also have observed a weird trend when testing between dataset of size 4000 and 40000 under 2 class non-IID circumstances. In other circumstances, training with larger dataset always result in higher accuracy, while in this circumstance the trend is inverted.

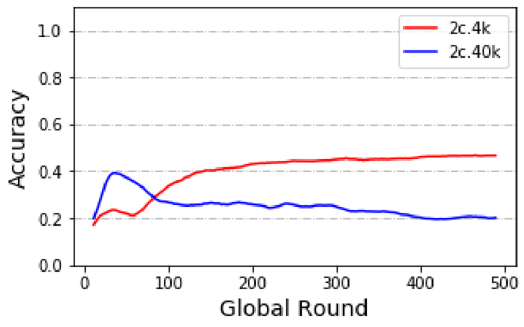


Figure 4. Test record of training with dataset of size 4000 and 40000 under 2 class non-IID circumstance. Manipulations are implemented on windowed accuracy and loss (window size equals to 21).

3.4.3 Default Simulation Settings

We select a balanced total training set of size 4000 or 40000 and apply evaluation with CIFAR10's whole testing set. Training set are distributed to 10 or 40 nodes depending on the experimental design. Client nodes are in equal size and samples from available classes within a node are balanced. By default, local epochs are set to 1 and global rounds are set to 1000. Evaluation is performed by the central node. In each scenario, we change 1 parameter and compare it to the one without that change.

Scenarios are tested under 2 class, 5 class and IID circumstances with dataset of size 4000 (4k dataset) and under 2 class, 3 class, 5 class and IID circumstances with dataset of size 40000 (40k dataset). Early stopping of 500 rounds has been applied to testing with 4k dataset.

3.4.4 Model Poisoning

We test this scenario where there is an abnormal node that returns arbitrary weights at each global round. We split the training set to 10 nodes and added in an abnormal node who claims to be 1% or 10% as large as total training dataset.

Testing is performed with both 4k and 40k dataset. Additional tests of 3% abnormal node size are performed with 40k dataset. Other addition tests of poisoning frequency and poisoning data are performed with 40k dataset under IID condition.

3.4.5 Data Dispersion

In some tests we distribute our data to 10 clients, in other tests to 40 clients. In this scenario, we compare the training performance between 10 client and 40 client cases in default settings. Testing is performed with both 4k and 40k dataset.

3.4.6 Delayed Update

In this scenario, some nodes update at 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.

From the secone aspect we test various speed of delayed nodes: 20% of 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.

3.4.7 Share data strategy

In previous tests, 1 class non-IID never behaves better than random. A research suggested a data-sharing strategy that can improve the training performance over 1 class non-IID data (Zhao *et al.* 2018). The author assumes a small IID set (2.5% to 25% as large as total training dataset) can be published for pre-training the model or sharing between nodes.

In this scenario, we attempt to 3 different methods of using the shared data. Testing is performed with both 4k and 40k dataset. 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 resample an IID set from the pool. Method 3 will duplicate small classes in the pool to make the pool label balanced. Test is performed under 1 class non-IID circumstances with 40k dataset and 10 client nodes, dataset to be shared is as large as 2.5% of the total set. Tests with other non-IID or IID circumstances or testing or other sizes of shared set with the 4k dataset is also performed.

3.5 Cloud Implementation

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 request to the server by every round. The server fixes its client number to 3, aggregation will start when 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 save the weight,
1. Server receive requests,
2. Server register clients and send weights,
3. (Clients perform local training),

4. Server receive returned weights, wait until 3 weights is ready,
5. Server aggregates weights and perform evaluation.

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

4 Results

4.1 Scenarios testing

In an ideal training process, best accuracy and minimal loss will be achieved at convergence epoch. In our trainings, we have observed 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 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 perform best at the global round with minimal loss, the accuracy at that round represents model's best performance. In our following plots, we will label this point by a colored square.

Since the original results are fluctuated, 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 accuracy and loss curve for both training and testing dataset. Also, we will append the windewed results at the end of the report

4.1.1 Model Poisoning

The presence of abnormal node leads to performance degradation. The influence is related to both the size of the abnormal node and the poisoning frequency. We label the situation when an abnormal node which is as large as x% of the total training set and upload arbitrary weight by every y rounds as ‘x%-y’. We observe similar accuracy trends among situation ‘1%-1’, ‘3%-3’ and ‘10%-10’, 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 abnormal node size per round. The influence of abnormal node size in this case might still exists. Loss of ‘10%-10’ stop decreasing at earlier rounds than ’1%-1’.

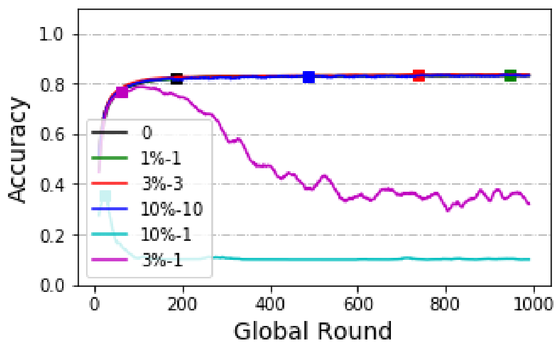


Figure 5. Accuracy curve of model poisoning tested with 40k dataset under IID circumstance. X axis represents global rounds and Y axis represents accuracy (window size equals to 21). '0' represents no poisoning to the system.

Models trained with large dataset seems to have higher tolerance to attacks from abnormal nodes. In the case of 1% size abnormal 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 slight decrease in non-IID groups (while there is no decrease in the IID group), the decrease does not appear much serious in 2 class than 5 class non-IID.

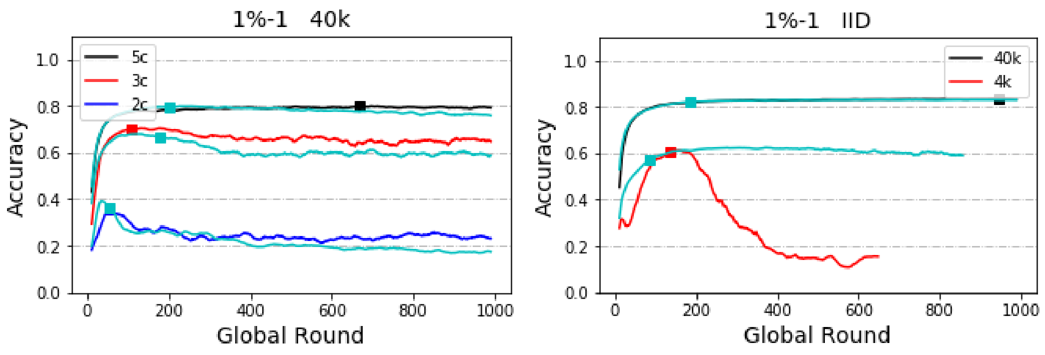


Figure 6. Accuracy curve of model poisoning. X axis represents global rounds and Y axis represents accuracy (window size equals to 21). The light blue line without label is the original curve without any poisoning for each case. Left: Poisoned by abnormal node of size '1%' by every round with 40k dataset, comparing between non-IID classes. Right: Poisoned by abnormal node of size '1%' by every round under IID circumstance, comparing between 4k and 40k dataset.

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

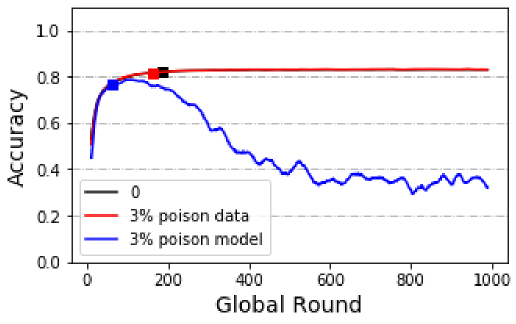


Figure 7. Accuracy curve of data poisoning and model poisoning. X axis represents global rounds and Y axis represents accuracy (window size equals to 21). Poisoned by abnormal node of size '3%' by every round, tested with 40k dataset and under IID condition.

4.1.2 Data Dispersion

From Figure 8 below we can see training with 40 client nodes is obviously 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 speeds (p-value=0.0059). Differences in accuracy at minimal loss is not significant (p-value=0.71).

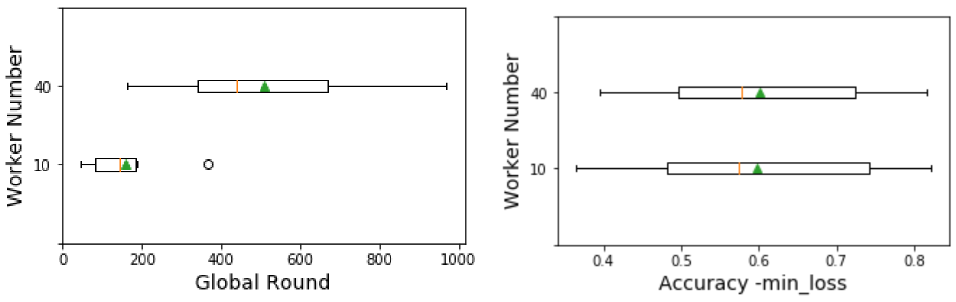


Figure 8. Box plot of round index (Left) and accuracy (right) where loss is minimum, group by worker number, including results tested with 4k and 40k dataset under various IIN and non-IID circumstances. Manipulations are performed on windowed testing accuracy and loss (window size equals to 21).

4.1.3 Delayed Update

Statistical tests are applied to detect the difference between no-delay-node case and other situations. Statistical tests show a significance between with or without delayed update in the speed to reach to 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 accuracy at minimal loss.

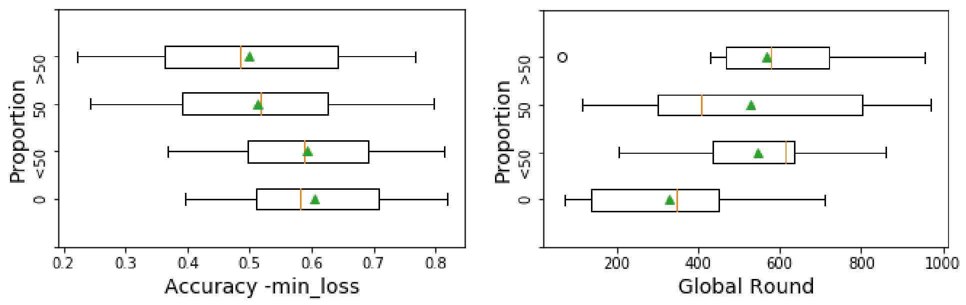


Figure 9. Box plot of round index (Left) and accuracy (right) where loss is minimum, group by proportion of delayed nodes, including results for delay speed equals to 3, tested with 4k and 40k dataset and under various non-IID or IID conditions. Manipulations are performed on windowed testing accuracy and loss (window size equals to 21). '0' means no delayed nodes in the system. '<50' represents delay proportion of 20% and 25% (for 4k and 40k dataset, respectively). '50' represents delay proportion of 50%. '>50' represents delay proportion of 75% and 80% (for 40k and 4k dataset, respectively).

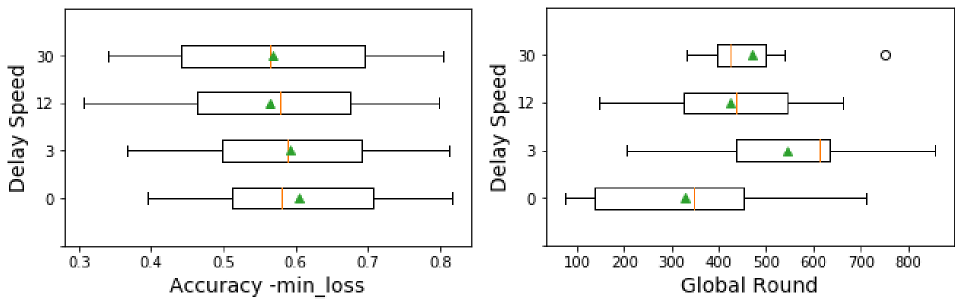


Figure 10. Box plot of round index (Left) and accuracy (right) where loss is minimum, group by delay speed, including results with 4k and 40k dataset when proportion equals to 20% or 25% (for 4k and 40k dataset, respectively). Tested under various non-IID or IID conditions. Manipulations are performed on windowed testing accuracy and loss (window size equals to 21).

4.1.4 Share data strategy

In this scenario we compare different methods of using the 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 total dataset, tested with 40k dataset under 1 class and 5 class non-IID. Both methods perform better than without sharing but still worse than IID case and are worse than without sharing case when the original data is not seriously non-IID (while in this case resample further worsen the performance). The direct method performs better than the resample method in most cases. But when the model is over-trained, in 1 class non-IID scenario (the local data pool is still very unbalanced when sharing only 2.5% of total data), accuracy of direct method decreases sharply after overfitting while accuracy of the resample method can stay stable. Testing results for sharing more data (tested with 4k dataset and under 1 class non-IID) also behaves similarly as sharing 2.5% under slightly non-IID cases (result in Appendix).

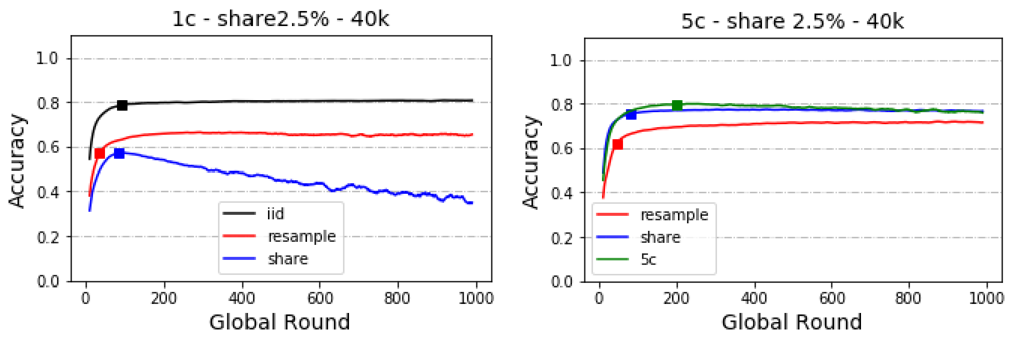


Figure 11. Accuracy curve of share data strategies compared to IID curve or original curve. X axis represents global rounds and Y axis represents accuracy (window size equals to 21), testing with 40k training set and 2.5 % shared set. Left: Results for 1 class non-IID circumstance compared to IID curve. Right: Results for 5 class non-IID circumstance compared to original curve.

Figure 12 shows the result of replicating the small classes in the pool to make labels balanced. This method performs even worse than the other two methods.

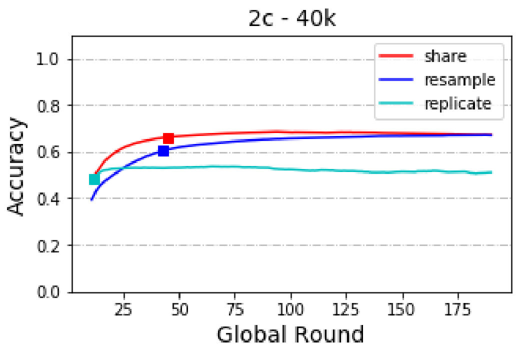


Figure 12. Accuracy curve of share data strategies testing with 40k training set and 2.5 % shared set under 1 class non-IID circumstance for 200 epochs. X axis represents global rounds and Y axis represents accuracy (window size equals to 21).

4.2 Cloud implementation

We have successfully performed our example task on the cloud. The result shows that 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 13 we zoomed the Y axis to show much clearer difference, the results in original scale are available in the Appendix.

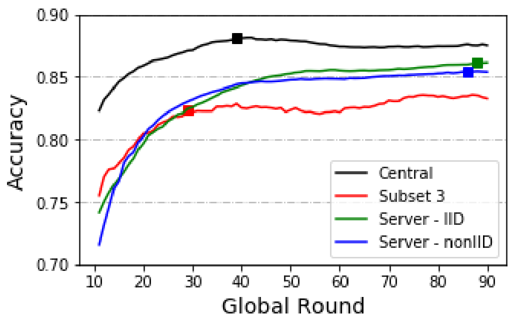


Figure 13. Accuracy curve of implementations with OIA-DDR dataset for 200 epochs. X axis represents global rounds and Y axis represents accuracy (window size equals to 21).

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 larger dataset and 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 will not be heavily affected in slight or low frequency attacks.

In testing with delayed update, we have observed the system to acting slower when delay happened. This slowering seems to be reduced when delay speed increase. A possible explanation of this is that delayed nodes of larger speed upload its weights at ower frequency than others. Due to the ambiguous statistical results, we are still uncertain about 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 in 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 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. But this opinion needs more testing. Our current results suggest that the majority of nodes should be synchronized when designing the federated system.

In the test with data dispersion, we detect no decrease in accuracy, thereby increasing the confidence of inviting small clients.

In the test with share data strategy, only when the training data is seriously non-IID, we found that the performance can be improved by using shared data. We suggest using share-only strategy when we don't know the distribution of data, since it can improve the situation in seriously non-IID cases while doesn't decrease performance for too much in slightly or no non-IID cases. But we can also use the resample strategy when we already know that shared amount is relevantly small and the distribution is 1 class non-IID, since the collapse of share-only method will be rapid and unpredictable. In any case, we should not duplicate the label since this method performs the worst.

We also have thought about shifting evaluation methods to the clients. Ideally, if the clients can split part of local data as 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 workers. There might be some limitation in transporting size, though usually deep learning models will not be that large. Another student in the Lab is focusing on communication research.

6 Conclusion

We have tested several scenarios in federated learning under various circumstances, the results can provide reference to future federated designs. Our simulation system can also be edited for more 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 bioinformatics or other people who would like to attempt on federate deep learning.

7 Acknowledgement

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

8 References

Angermueller C, Pärnamaa 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:

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.

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

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

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.

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

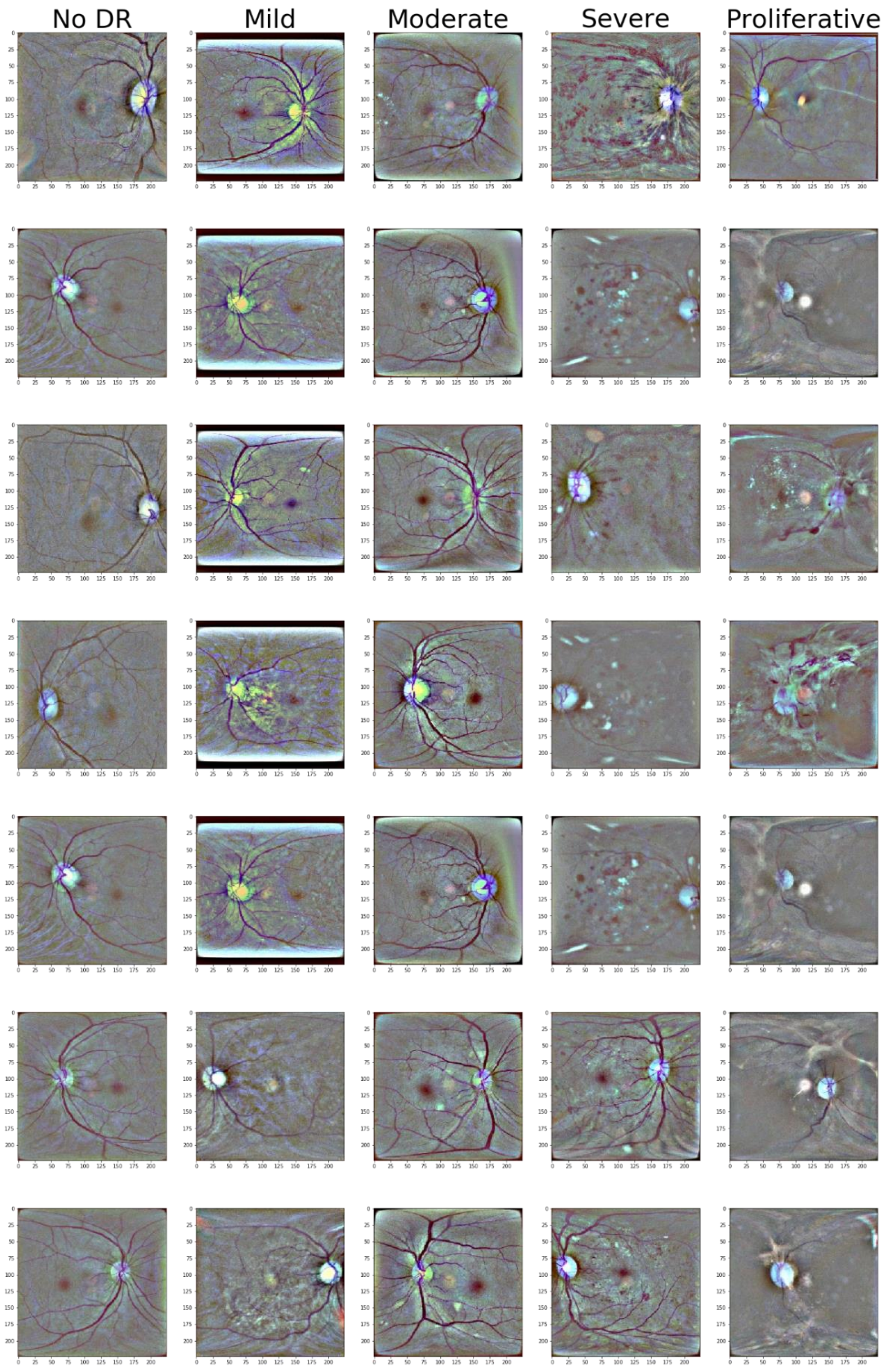
Stokes JM, Yang K, Swanson K, Jin W, Cubillos-Ruiz A, Donghia NM, MacNair CR, French S, Carfrae LA, Bloom-Ackermann Z, Tran VM, Chiappino-Pepe A, Badran AH, Andrews IW, Chory EJ, Church GM, Brown ED, Jaakkola TS, Barzilay R, Collins JJ. 2020. A Deep Learning Approach to Antibiotic Discovery. Cell 180: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

Yang Q, Liu Y, Chen T, Tong Y. 2019. Federated Machine Learning: Concept and Applications. 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

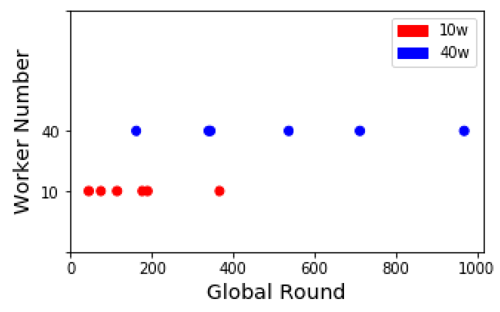
9 Appendix - OIA-DDR dataset after pre-processing

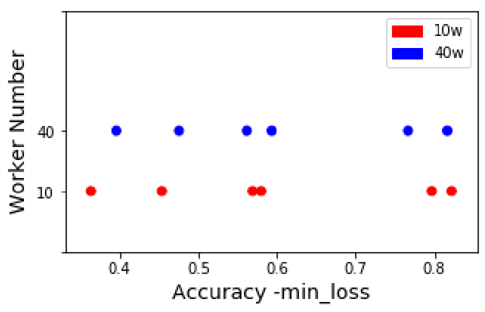
Methods from (https://www.kaggle.com/ratthachat/aptos-eye-preprocessing-in-diabetic-retinopathy) and (https://www.kaggle.com/titericz/circle-to-rectagle-preprocessing-1) are used for proprocessing original images to our input data.



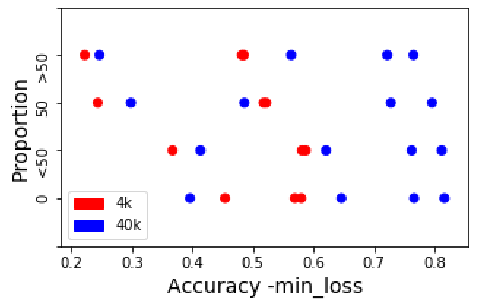
10 Appendix – Samples used in Box plots

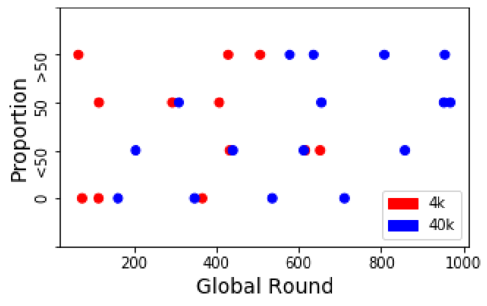
Data Dispersion

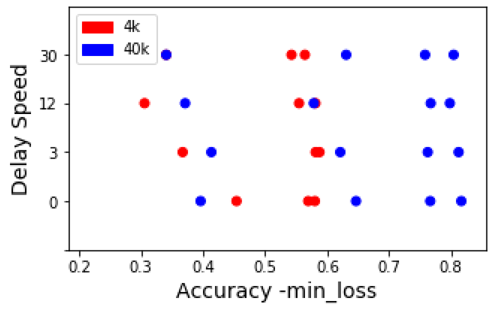


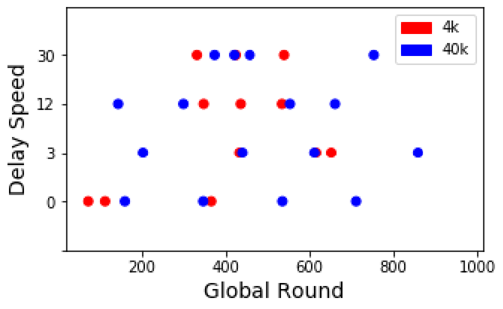


Delayed Update









11 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 T-test. We perform T-test or Wilcoxon test depending on assess results. Results tested by Wilcoxon test are tagged with ‘#’.

Data Dispersion

	‘10W’-‘40W’
ACC	0.71
ROUND	0.0059 *

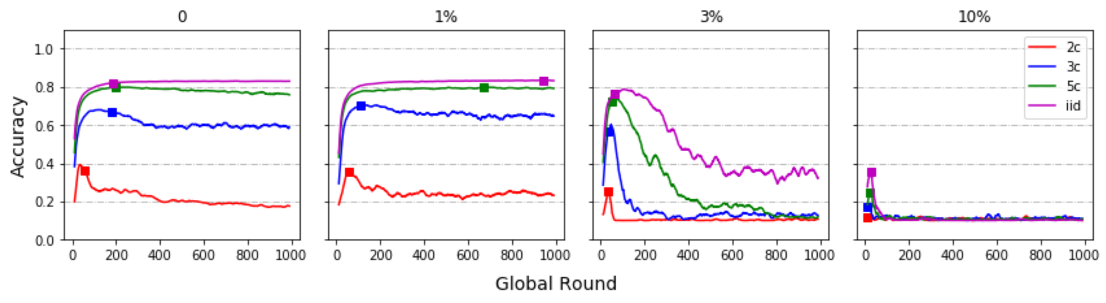
Delayed Update

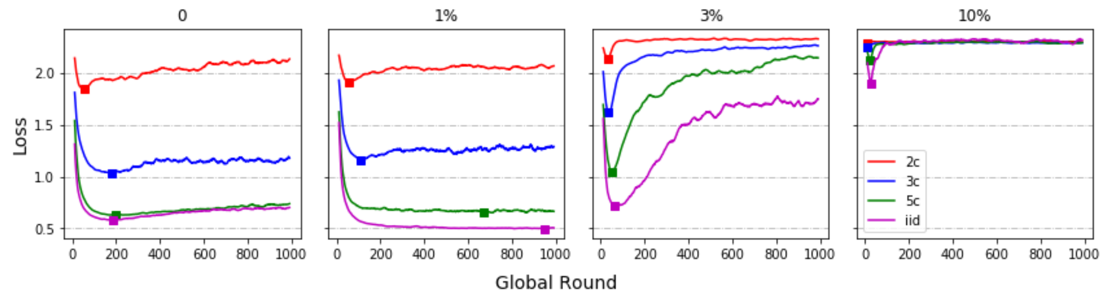
PROPORTION	**‘0’-‘<50’**	**‘<50’-‘50’**	**‘50’-‘>50’**	**‘0’-‘50’**	**‘0’-‘>50’**	**ANOVA**
ACC	0.41	0.0041 *	0.39	0.01 *	0.005 *	0.62
ROUND	0.016 *	0.90	0.60	0.089	0.092	0.35

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

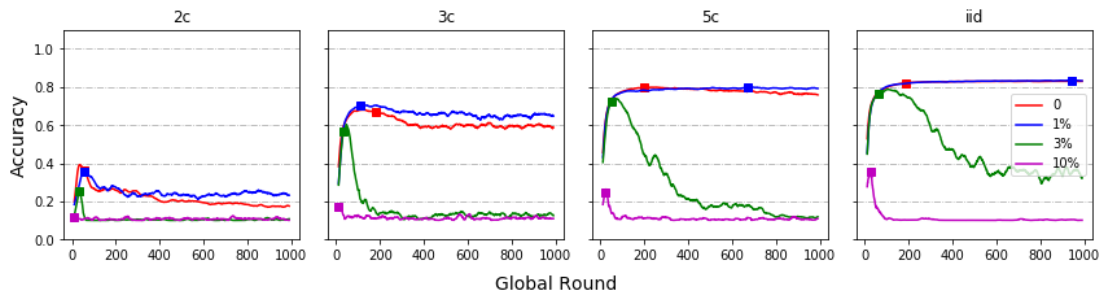
12 Appendix - Scenario testing results with 40k set

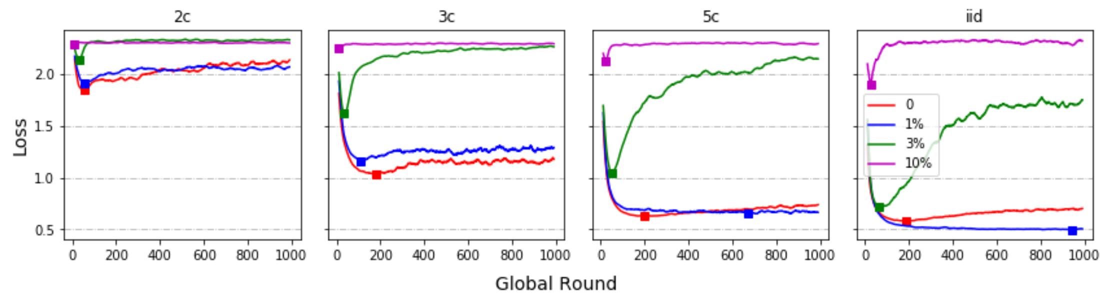
Model Poisoning – By Abnormal Node Size



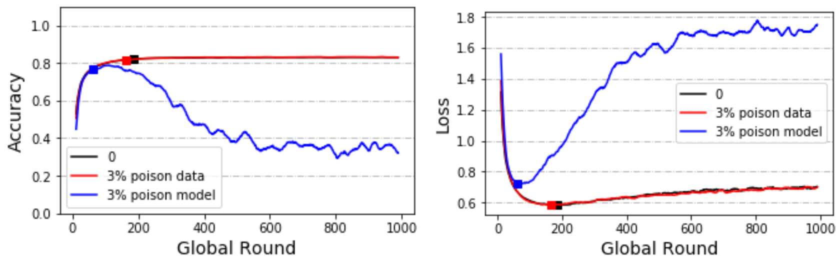


Model Poisoning – By IID/non-IID

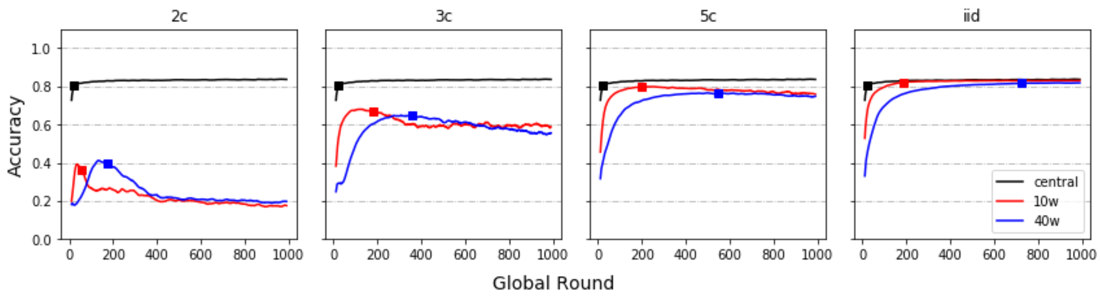


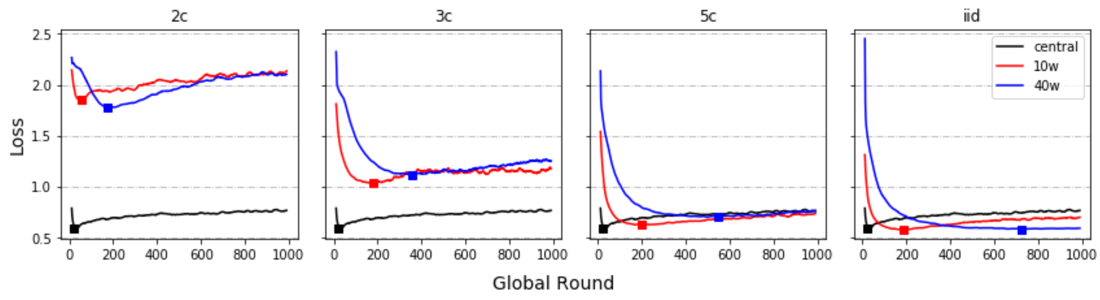


Data Poisoning

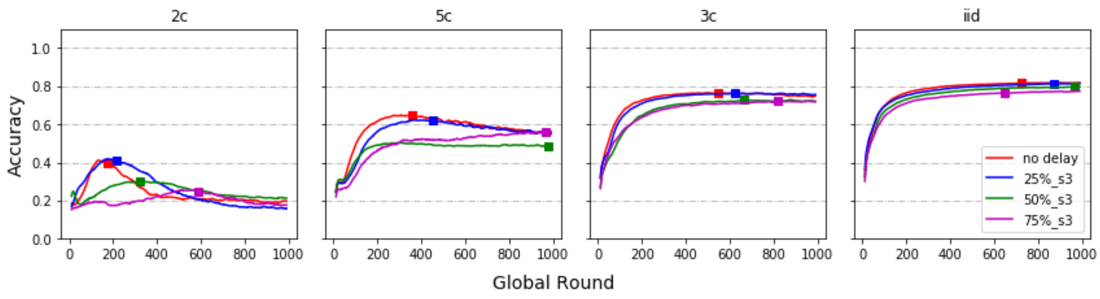


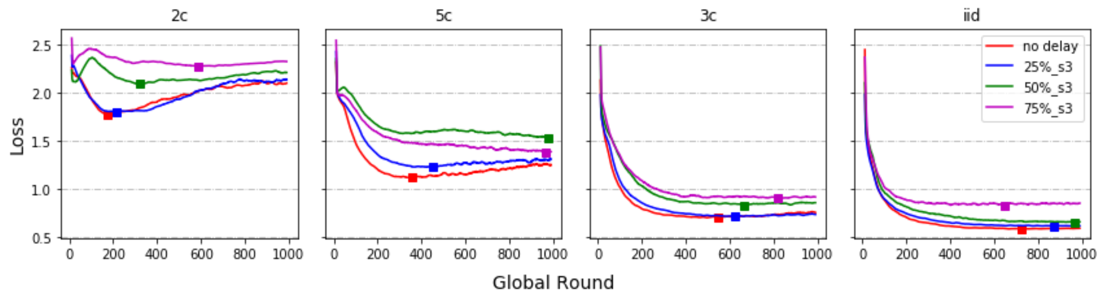
Data Dispersion



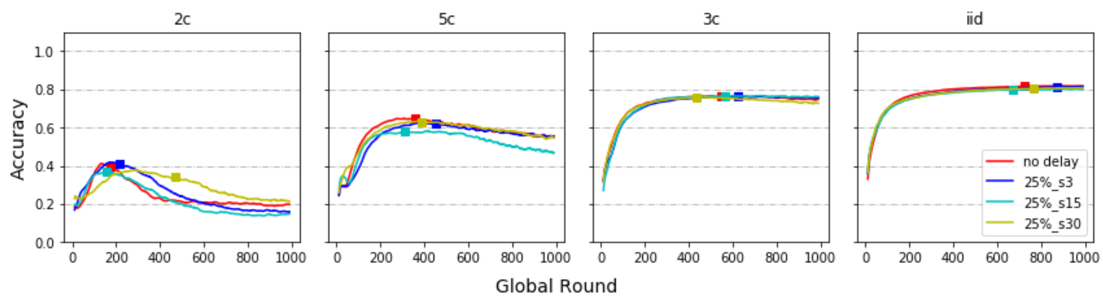


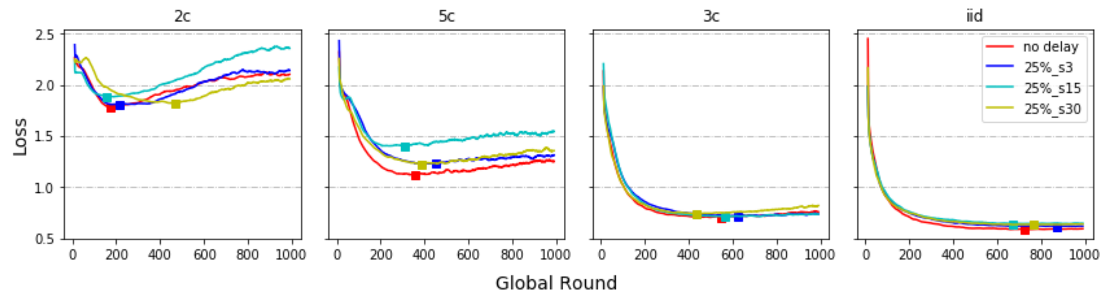
Delayed Update – By Proportion



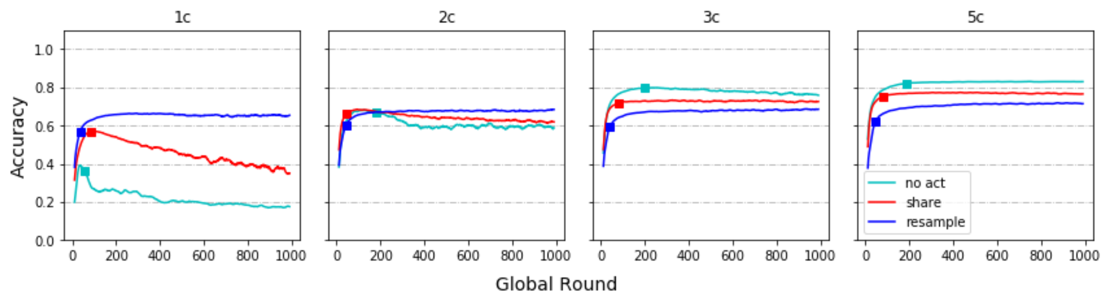


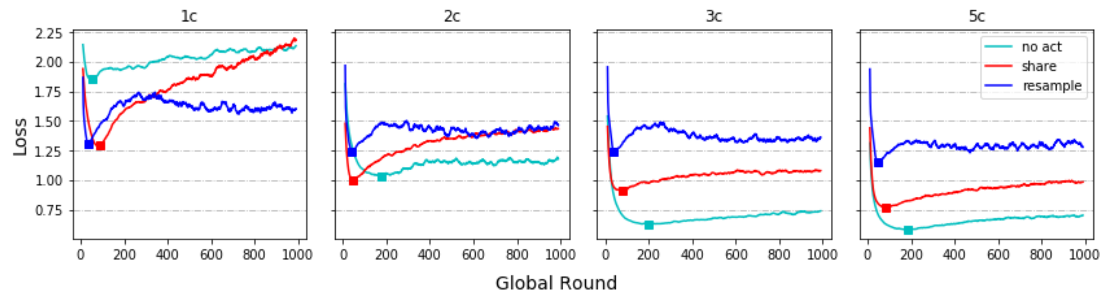
Delayed Update – By Speed

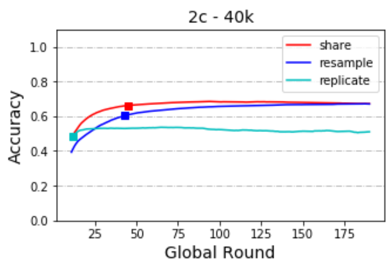


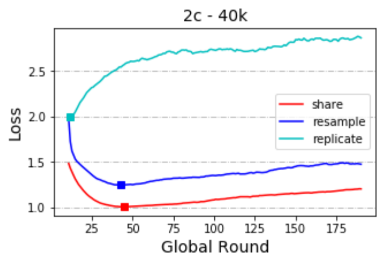


Share data



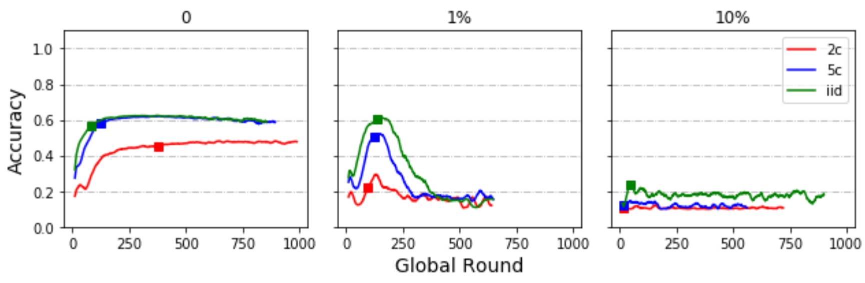


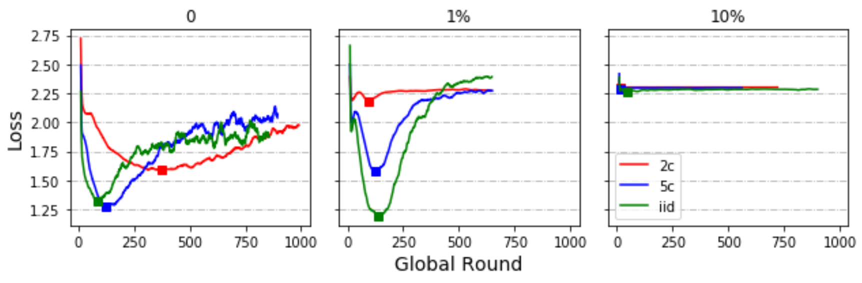




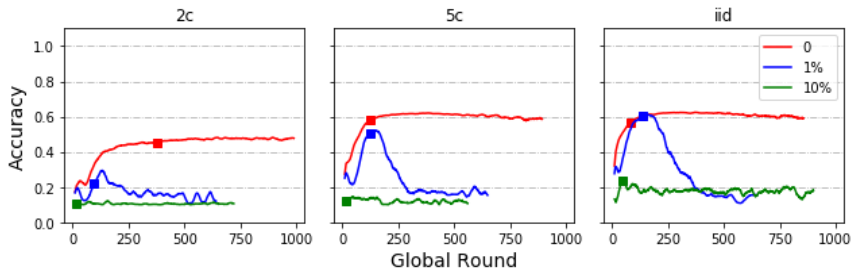
13 Appendix - Scenario testing results with 4k set

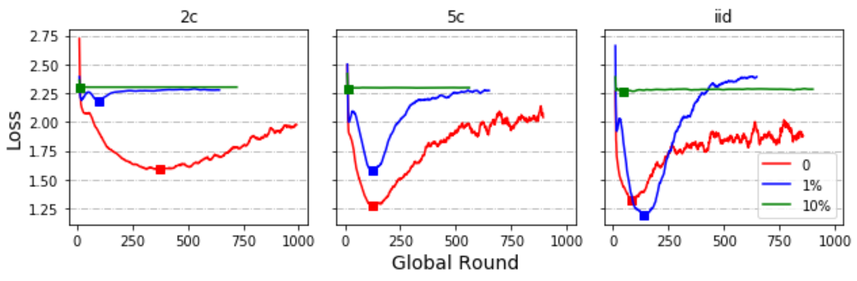
Model Poisoning – By Abnormal Node Size



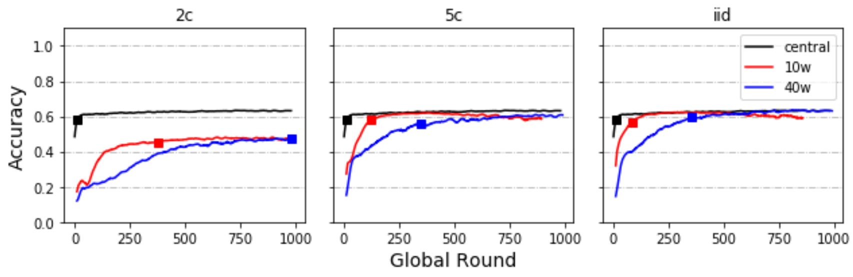


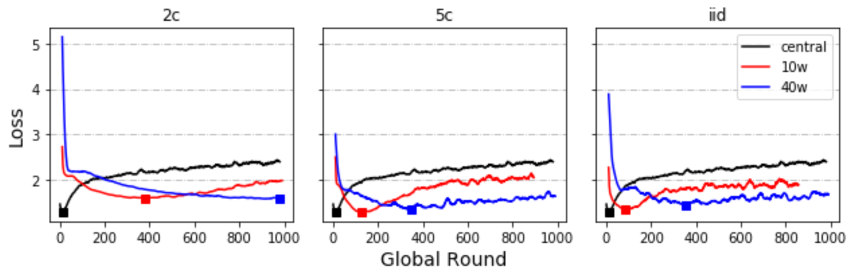
Model Poisoning – By IID/non-IID



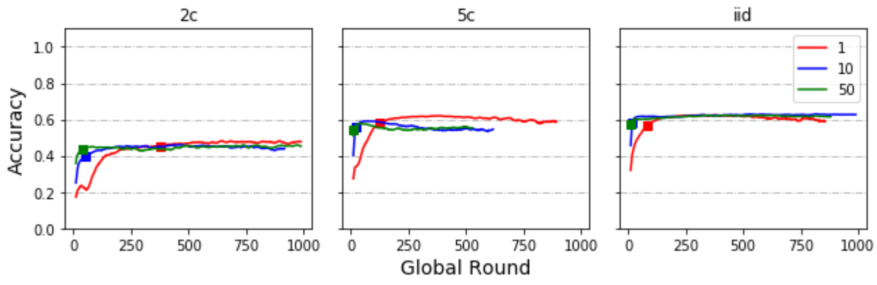


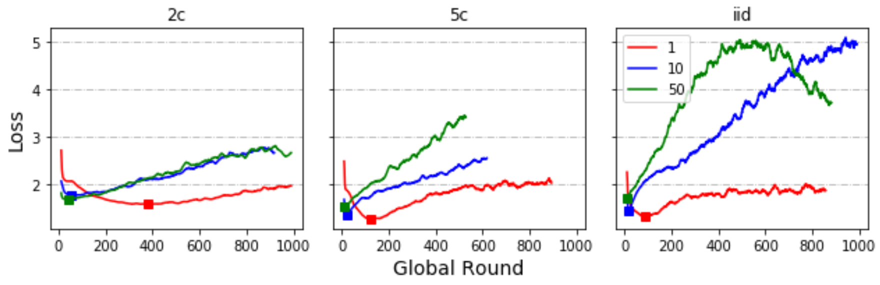
Data Dispersion



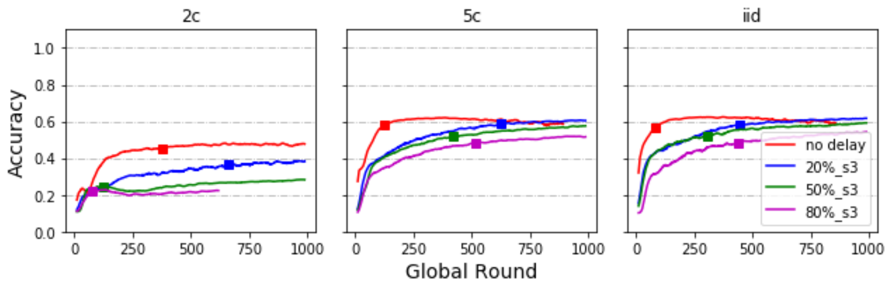


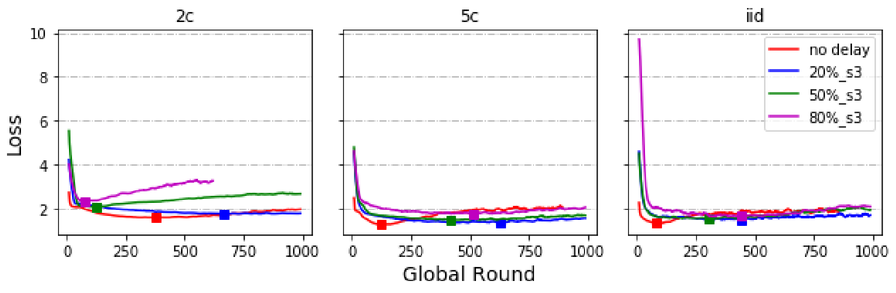
Local Round



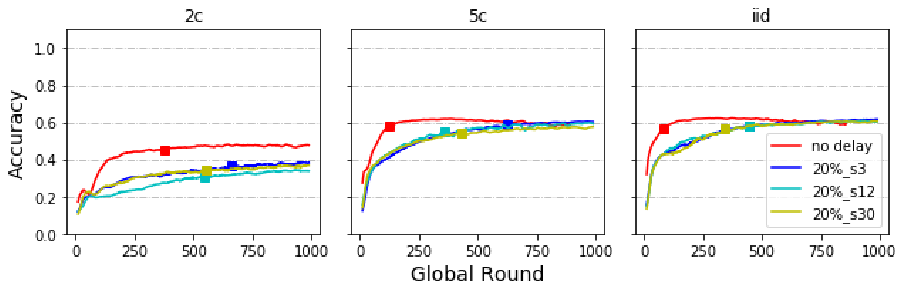


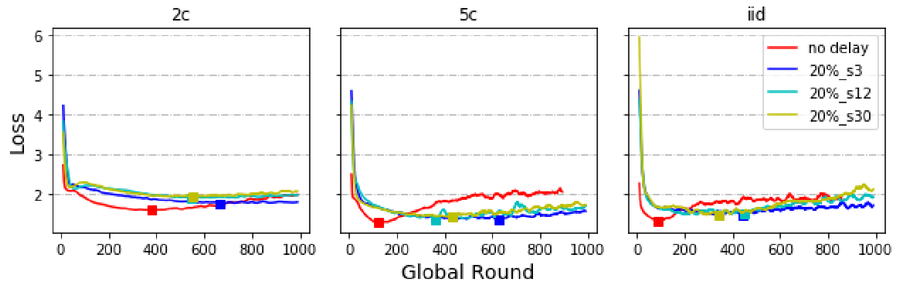
Delayed Update – By Proportion



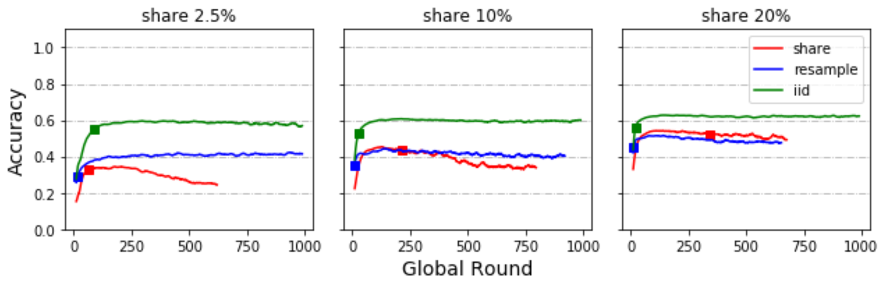


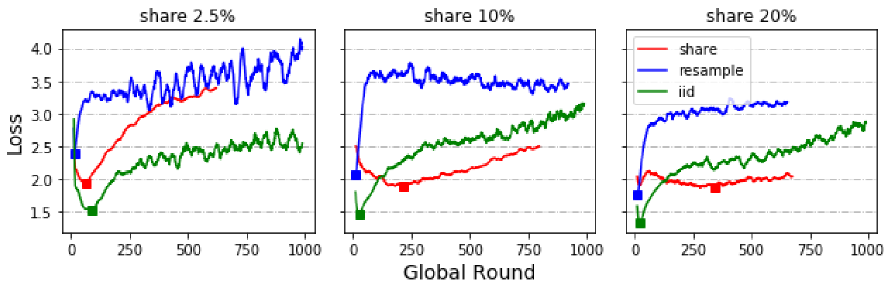
Delayed Update – By Speed



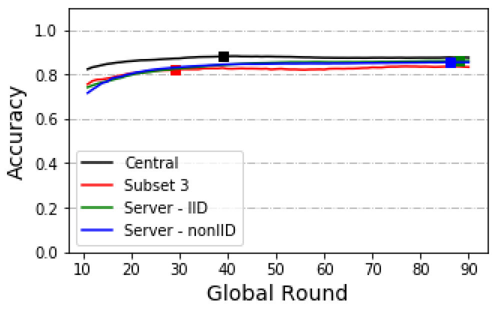


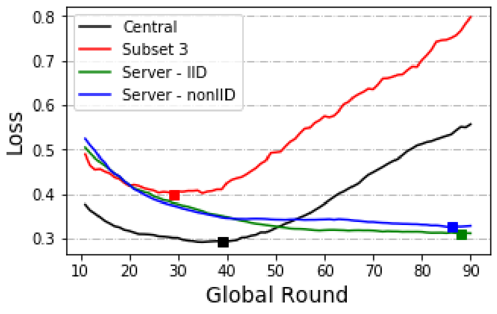
Share data




share 2.5%
share 10%
share 20%
Loss
Global Round
share
resample
iid
4.0
3.5
3.0
2.5
2.0
1.5
0
250
500
750
1000

14 Appendix - Results for cloud implementation


Accuracy
Global Round
Central
Subset 3
Server - IID
Server - nonIID


Loss
Global Round
Central
Subset 3
Server - IID
Server - nonIID