1. Introduction
2. Background

2.1 Machine Learning in Bioinformatic fields

What can be done using ML (several examples using Neural network), may other ways that use old machine learning.

2.2 Performance and dataset

If there’s a paper about data size and training performance… Introduce several famous datasets in biology/medical fields. But compare to large dataset/baseline like imagenet xxx. Or describe the need for dataset like lack sth???

2.3 Federated Learning

What is / why use FedML (privacy/distribute computing or storing resources), challenges, federated learning frameworks. Federated success case in biology field.

1. Datasets and Methods

In this project, the CIFAR-10 dataset has been used for locally simulating different scenarios of federated learning, the OIA-DDR dataset has been used for implementing an example bioinformatic federated learning on the cloud. The performance of the models will not be high enough for any actual usage, only for testing differences among federated training scenarios and the centralized training.

We test 1000 epochs (for centralized training) or global rounds (for federated learning) for each simulation task and 100 epochs/rounds for the cloud implementation

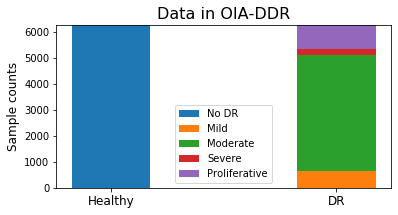
3.1 CIFAR-10 dataset

The CIFAR-10 dataset consists of 50,000 training images and 10,000 testing images, each image is in color format and the size is 32x32 pixels. In this project, 40000 randomly picked training images and the whole testing set have been used for scenario testing.

3.2 OIA-DDR dataset

The OIA-DDR dataset provides high-quality diabetic retinopathy 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. (<https://www.kaggle.com/ratthachat/aptos-eye-preprocessing-in-diabetic-retinopathy>) (<https://www.kaggle.com/titericz/circle-to-rectagle-preprocessing-1>)



(Figure xxx each class’s count)

3.3 Training designs

The task for CIFAR-10 is to input images of size 32x32x3 and classify them to 10 classes. The model is provided by Mattias Åkesson. We use ADAM as the optimizer with the default learning rate at 1e-3. The batch size is set to 100 (in formal implementation) or smaller (in pilot tests). Labels are in integer format; loss will be calculated using categorical cross entropy. Other parameters are using TensorFlow’s default settings.

The task for OIA-DDR dataset is to input images of size 224x224x3 and classify them to 2 classes. VGG16 network (pre-trained on ImageNet dataset) is used. We use ADAM with learning rate initialized at 1e-4 and apply inverse time decay by rate 0.05 in every epoch. The batch size is set to 300. Labels are in string format; loss will be calculated using binary cross entropy. Other parameters are using TensorFlow’s default settings.

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

In this project, we evaluate the training performance with accuracy and loss.

3.4 Scenarios testing

We simulate and test different scenarios in federated learning in this part. In this simulation system, we assume a folder to be the central node and several folders to be worker nodes and assign the index of training data to each worker accordingly.

When the system starts, the central node will initialize and save the model weight in its folder. We apply Federated Averaging in this system, weighted by node’s size. Each epoch consists of the following basic actions:

1. Worker nodes generate local model,
2. Worker nodes load weight from the central node,
3. Worker nodes train the model over their local data and save the history,
4. Worker nodes return weights to the central nodes,
5. Central node averages the weights and evaluate on the testing set,
6. Central node saves weights in its folder.

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. We will show the windowed accuracy and loss of our results, since the original accuracy are fluctuated. The original plots can be seen in the appendix.

Before the formal implementation of different scenarios, we first test the system with smaller dataset and different non-IID classes, the results can be found in the appendix. Here is a plot showing how centralized and federated learning of our model behaves when having different size of training dataset. The plot shows the windowed accuracy of the global round with minimum windowed loss of the first 500 epochs. The result fluctuates due to randomness, but it is obvious that the performance of both federated and centralized learning can be improved by larger training set, and federated learning converge at slower speed than the centralized learning.

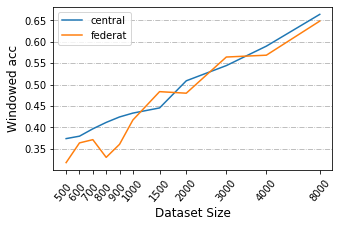
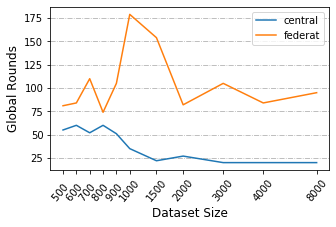
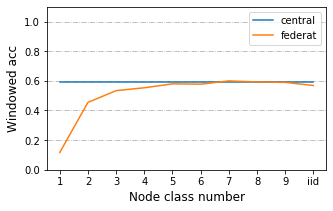
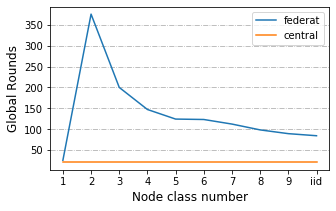
 

Figure xxx

LEFT: This figure shows the windowed accuracy of the global round with minimum windowed loss for different dataset size of the first 500 epochs (window size=20). RIGHT: This figure shows the global rounds to reach to minimum windowed loss.

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, so we show it by identifying how many classes each node has. Here’s a plot showing our model’s behavior at different non-IID scenarios when using 4000 images as training set:

(Figure xxx. Pilot test)

LEFT: This figure shows the windowed accuracy of the global round with minimum windowed loss for different non-IID classes of the first 500 epochs (window size=20).

RIGHT: This figure shows the global rounds to reach to minimum windowed loss for different non-IID classes of the first 500 epochs (window size=20).

In this case, 1 class non-IID never behaves better than random, 2-4 classes non-IID behave obviously worse than central training, 5-9 classes non-IID and the IID classes can reach similar accuracy as central training. Seriously non-IID classes converge at slower speed than IIN classes. Since we are not sure if different non-IID degrees will behave in similar trends under different scenarios, 1 class, 2 classes, 5 classes and IID classes are tested for the following scenarios. We neglect 1 class non-IID in most scenarios since we know it will never enhance performance through local training.

***Default Simulation Settings***

By default, we will have 40000 images as total training set and 10000 images as total testing set. Labels in each set are well balanced. We will split the training set to 10 or 40 nodes depending on the experiment design. In each test the size of the worker nodes will be equal and the sample size of available classes within a node will be equal. Local rounds are set to 1, global rounds are set to 1000. Evaluation is done by the central node.

***Abnormal node***

In this scenario we test the situation where there is an abnormal node that returns randomized weights at each global round. We split the training set to 10 nodes and added in an abnormal node who claims to have 40 or 400 training data. We test this scenario with 2 classes, 5 classes and IID classes.

***Local Rounds***

In this scenario, we test with 10 local rounds before uploading. A previous research has tested this scenario with small local steps on 1 and 2 class non-IID dataset (Zhao *et al.* 2018), here we test this effect with 2 classes, 5 classes and IID classes. We split the training set to 10 nodes.

***Data Dispersion***

In default settings, we distribute the training set to 10 nodes. In this scenario we split the training set to 40 nodes.

***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:

1. Different proportion of delayed nodes: 25%, 50% or 75% nodes 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.
2. Different speed of delayed nodes: 20% nodes delayed, 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.

***Share data***

This scenario is tested with 1 class, 2 classes, 5 classes and IID classes, we split the training set to 10 nodes. In pilot tests, 1 class non-IID never behaves better than random. A previous 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.

We apply this strategy to our model with sharing size equals to 2.5% of the total training set. In each node there will be a mixture of shared data and local data. In our design, an IID local training set is randomly resampled from the local sample pool. We compare the result of our design to the original strategy. In order to simplify the pipelines, pre-training step is not applied.

3.5 Cloud implementation (NOT yet finished)

This

1. Results

4.1 Scenarios testing

***Abnormal node***

***Local Rounds***

***Data Dispersion***

***Delayed Update – Proportion’s effect***

***Delayed Update – Speed’s effect***

***Share data***

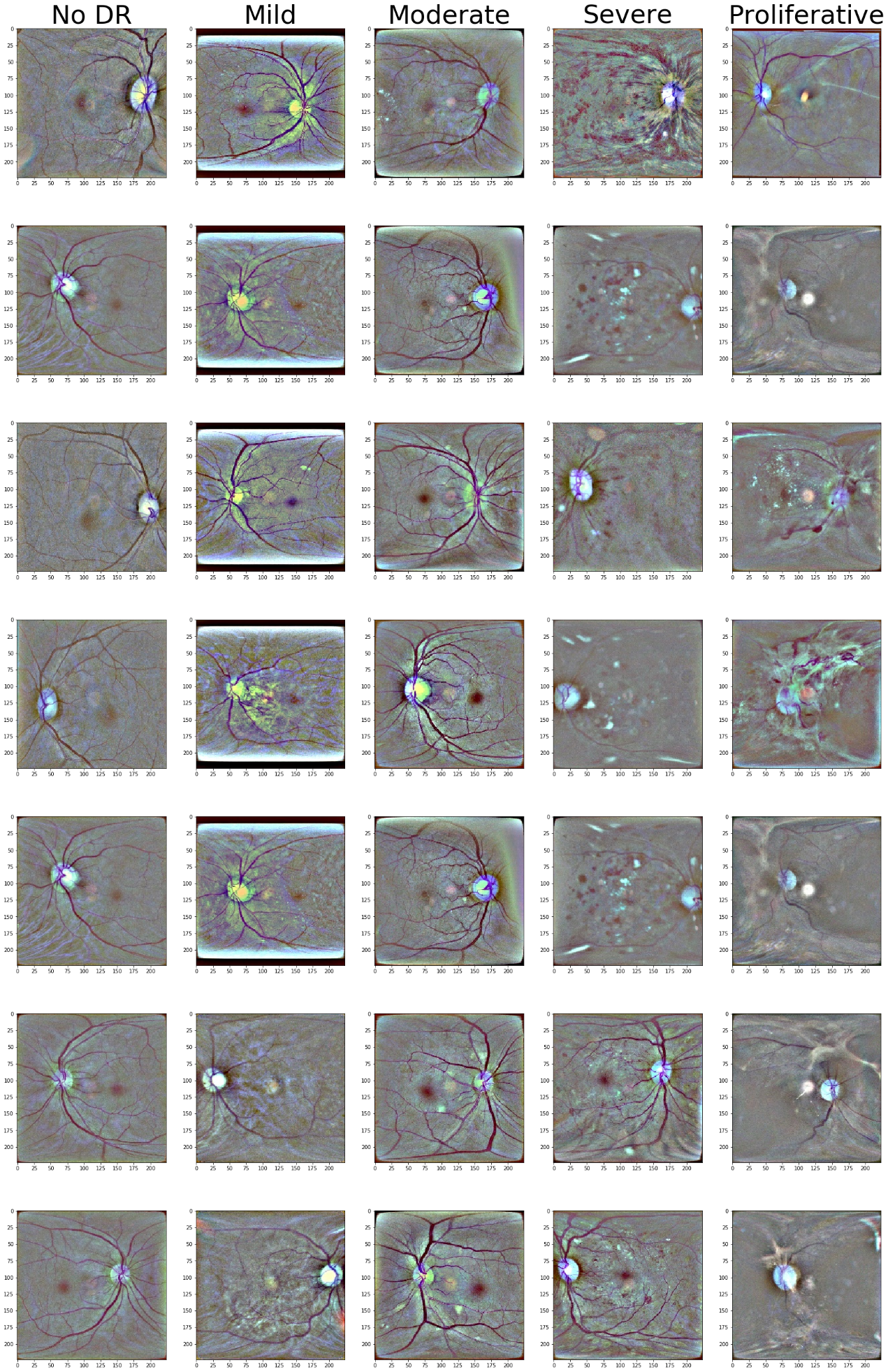
4.2 Cloud implementation

1. Discussion

(like how to evaluate on node.)

1. Conclusion
2. Acknowledgement
3. Reference

Appendix 1 OIA-DDR dataset after preprocessing



Appendix 2