Reproduce Distributed Learning Networks for Medical Imaging and Investigate the Performance in Edge Scenarios (Healthcare)

Miao Zhang
Stanford University
miaoz18@stanford.edu

Abstract

How to organize data is a challenge for large scale machine learning, especially in healthcare area where privacy and regulation of data sharing need to be addressed carefully. Recently techniques, such as Federated Learning, aim to build incentive models on distributed and unstructured data for data transparent ecosystems. I implemented the technique shown in the paper[1] that trains the deep neural network on 4 separate client sites with no data centralization but only the communication of model weights, and obtained the result with test accuracy around 74%. Furthermore, I extended the work in the paper by investigating the difficult scenarios that happens in real life including data quality and class imbalance.

1 Introduction

Deep learning has been widely applied for clinical diagnosis, and usually collaboration among multiple institutions is necessary to achieve high algorithm performance when data is limited. However, medical data often has limitations to be shared or published due to legal, or ethical concerns. Therefore, researchers have explored methods of distributing deep learning models as an alternative to sharing patient data, meanwhile achieving high algorithm performance. One effective approach is model averaging, where separate models are trained for each split of the data and the weights of the model are averaged every few mini-batches to the server. The work[1] explored this approach and compared it with normal centrally hosted data learning. Although the results of distributed learning is not as good as centrally hosted learning, which is expected, it still showed satisfying performance and proved that this technique is promising.

In the first part of this work, I reproduced the model and algorithm proposed in the paper[1] based on the newly released Tensorflow Federated (TTF) open-source framework[6]. The input to the deep learning model are human retinopathy eye images and the output will be the binary results which classify the images to "healthy" or "sick". Non-parallel training heuristics was used to simulate the distribution of networks across institutions. In the second part, I assessed the questions: (1) how one or multiple institutions having low-resolution images influence the performance of distributed learning, (2) how one or multiple institutions having patients data with class imbalance influence the distributed learning, and proposed possible improvement algorithms to these two edge cases.

2 Dataset and Preprocessing

The dataset used for training and evaluating the distributed networks is Kaggle Diabetic Retinopathy Detection Dataset, which is public. The data was preprocessed via the method detailed in the

,

competition report[4]. To summarize, I used the OpenCV Python package to rescale images to a radius of 300, followed by local color averaging and image clipping. The images were then resized to 256×256 to reduce the memory requirements for training the neural network. As a two-class logistic regression question, images were labeled in the way shown in the paper[1], which is 0(healthy) and 1(diseased).

There are 5712 training examples, 1500 validation examples and 1500 test examples in total. The size is similar to the dataset used in paper[1], and the difference is that they distributed 1500 patients for training in each institution, 3000 patients for validation, and 3000 patients for testing. However, this setup got worse performance than mine in my experiments. Since the training examples were divided into 4 distributed clients to train local models, to prevent overfitting and to improve learning, I used horizontal flip data augmentation flips (50% change of horizontal or vertical) of the images at every epoch, which was also applied in paper[1].

3 Methods

For the local learning part: I am using a 34-layer residual network (ResNet34) architecture pretrained on ImageNet to do the images classification, shown in Figure 1. The input shape was (256, 256, 3) and the output layer was a 2-class sigmoid layer. The weights of the network were optimized via a stochastic gradient descent algorithm with a mini-batch size of 32. The objective function used was binary cross-entropy. The learning rate was set to 0.001 and momentum coefficient of 0.9. All of these are the same as what illustrated in paper[1], and the only difference is that they adjusted the learning rates by multiplying by 0.25 when the same training images were used to train the neural network 20 times with no improvement of the validation loss. However, I can't adjust the learning rates dynamically because of the inner structure of TTF tool.

For the server averaging part: In each averaging round, the server model weights were distributed to 4 local institutions to train a model for a predetermined number of epochs, then the server collected and averaged the output model weights of all institutions and updated the server model state. By default it performs arithmetic mean aggregation, weighted by a function parameter. The server model was evaluated after each averaging round and tested after the final round by calculating the classification accuracy on the unseen testing cohort.

To investigate the edge scenarios, I introduced variability to 1 institution by decreasing the resolution of the images by a factor of 10, and class imbalance (ratio of healthy to diseased) was 10:1. In the same way, the variability was introduced to 3 institutions. The alternatives of skipping or not skipping the variable intuitions were assessed. Besides, to avoid the data deficiency impact from skipping multiple institutions, another approach of adjusting the weights for local outputs when averaging was also assessed to tackle with the variability.

The codes of the project can be viewed at: https://github.com/zm17943/CS230-Winter2020-FinalProject.

4 Experiments, Results, and Discussion

Using the weight averaging heuristic, the model was firstly initialized at the server and was transferred to 4 institutions. Local models were trained at each institution for 1 epoch and their weights were transferred back to the server to be averaged, and this is referred as an "averaging round". After the server model updating its weights, it was distributed to institutions again. The experiments results showed decreasing loss over averaging rounds, and after 5 averaging rounds, the plateau of validation loss was reached. Table 1 shows the accuracy results:

After comparing with the results from the paper[1], I found that the best result from the paper (76.8%) is slightly better than the results I got (74.1%). There are two possible reasons contributing to the difference in accuracy: the first one is that I didn't perform the adjustment on the learning rates during the training, which might prevent the model from converging to the optimal point. Especially in this situation that the training data size is not very large, the learning rate is supposed to be taken care of because it is observed from the table that the models quickly fit on the training set but seemed to have not very good generalization on the validation and testing set. The second one is the strategy of averaging batch norm layers in the model. ResNet models highly rely on the batch normalization,

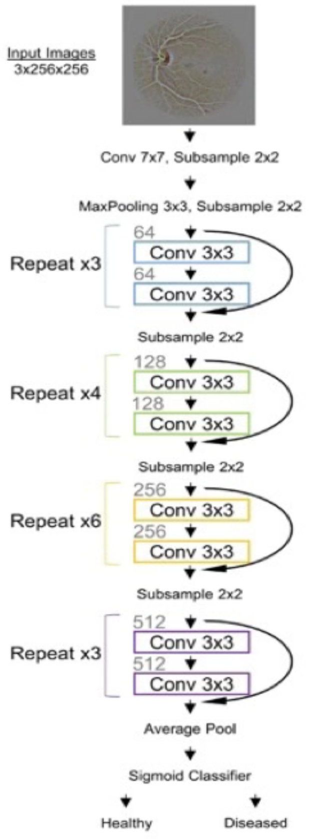

Figure1. ResNet-34 architecture utilized for the Diabetic Retinopathy dataset[1].

Rounds	Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)
1	61.1	59.9	58.8
2	76.2	67.5	66.9
3	83.9	71.2	70.5
4	90.7	74.0	73.8
5	95.3	73.9	74.1

Table1. Training, validation, and testing accuracy of the neural network in each round with weights averaged by distributed networks.

but TTF leaves it an open question and simply sets the accumulating mean and variance in batch norm layers to 0 and 1 in the server model, which possibly inflects the performance. Paper[1] doesn't reveal the batch normalization averaging method it used.

In the initial division of the different institutions, it was assumed that each institution had the same number of patients, ratio of healthy to diseased patients, and image quality. To simulate the real scenario that variability exists, I introduced variability into one of the 4 institutions and assessed the performance of the distributed training heuristic. I simulated two scenarios: In the first, I decreased the resolution of the images by a factor of 16. In the second, I introduced class imbalance (ratio of healthy to diseased was 9:1). The training heuristic is still the weights averaging. To explore the

	Testing accuracy (%) (Image quality variability)	Testing accuracy (%) (Class imbalance variability)
Skip the variable institution	72.9	72.9
Not skip the variable institution	71.5	70.2

Table2. Testing accuracy of the distributed network after five averaging rounds with skipping and not skipping the variable institution.

	Testing accuracy (%) (Image quality variability)	Testing accuracy (%) (Class imbalance variability)
Non-weighted averaging	70.7	68.3
Institution skipping	62.6	62.6
Weighted averaging	70.9	70.1

Table3. Testing accuracy of the distributed network after five averaging rounds with non-weighted averaging, skipping the variable institution, and weighted averaging.

possible solution to the variability, I assessed the performance of not skipping vs skipping the variable institution entirely. The results after 5 averaging rounds are shown in Table 2.

It can be observed from Table 2 that introducing image quality and class imbalance variability to one institution would both result in a degraded performance of distributed learning, and class imbalance had a greater impact. The alternative of skipping the variable institution in averaging helped, but the performance(72.9%) is not as good as the 4 institutions averaging heuristic(74.1%), which is expected because the model didn't learn knowledge from the part of training data that was skipped. Therefore, in the deployment of distributed learning in real clinical applications, if the proportion of variable institutions is small, excluding them from the averaging could be a possible solution.

Here comes a question that if a large proportion of institutions have data variability, it is not feasible to exclude all of them. To explore the scenario where there is variability in multiple institutions, I introduced image quality, patients number and class imbalance variability to 3 institutions in the same way as described above. Instead of skipping these 3 institutions, I used a weighted average strategy, where weights from each institution were not treated equally, but proportional to the degree of the variability. In this case, the resolution of the images was decreased by a factor of 10, and class imbalance (ratio of healthy to diseased) was set to 10:1. Thus, the degree of variability was viewed as 10, and I weighted the outputs of the three variable institutions by 0.1 during model averaging. Table 3 shows the testing results of the model trained with skipping, non-weighted averaging and weighted averaging strategy.

It can be observed from Table 3 that introducing the variability to three institutions further decreased the model performance comparing with one-institution variability scenarios: For image quality variability, 71.5% to 70.7%. For class imbalance variability, 70.2% to 68.3%. The approach of skipping the variable institutions during averaging was proved to fail (62.6%), which is possibly because the training set was decreased to the 0.25 of the original size so it is too small to fully train the model. The approach of weighted averaging can mitigate the variability-induced accuracy loss, and the improvement is more obvious in the class imbalance scenario (68.3% to 70.1%) than the image quality scenario(70.7% to 70.9%). Therefore, adjusting the averaging weights of institutions in distributed learning can be a promising strategy to be used in edge scenarios.

5 Conclusion

In this work, firstly, I implemented a distributed learning network consists of a server and four institutions. By prepossessing the data in the same way as illustrated in paper[1] and trained the model using distributed averaging heuristics, I reproduced a network which has the similar performance as that in paper[1] on medical imaging recognition tasks. Secondly, I conducted experiments to study the impact of data heterogeneity on distributed learning. It covers two heterogeneity scenarios: image quality variability and class imbalance variability, and two optimization strategies to overcome the performance loss: skip the variable institution during averaging and weighted averaging. The experiments show that: (1) Introducing data heterogeneity to institutions impacts the network performance, and the network is more vulnerable to the class imbalance variability than the image quality variability. (2) Skipping the institution with heterogeneity helps mitigate the performance loss in scenarios where only a small number of institutions have data heterogeneity. (3) Weighted averaging is an attractive alternative to mitigate performance loss of multiple institutions heterogeneity scenarios.

The analysis in this report proposed the idea of weighted averaging method, but to explore the optimal weighted strategy under different scenarios, more experiments should be performed afterwards. The future work could also include more detailed analysis for the deployment of distributed learning in real-world medical imaging settings, for example, to investigate mores types of edge scenarios and better distributed learning methods.

References

[1] Chang, Ken, et al. "Distributed deep learning networks among institutions for medical imaging." Journal of the American Medical Informatics Association 25.8 (2018): 945-954.

[2] McMahan, H. Brendan, et al. "Communication-efficient learning of deep networks from decentralized data." arXiv preprint arXiv:1602.05629 (2016).

[3] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[4] Kaggle. Diabetic Retinopathy Detection. 2015. https://www.kaggle.com/ c/diabetic-retinopathy-detection, Accessed April 1, 2017

[5] Graham B. Kaggle Diabetic Retinopathy Detection Competition Report. 2015. https://www.kaggle.com/c/diabetic-retinopathy-detection/discussion/15801

[6] Tensorflow Federated. https://www.tensorflow.org/federated

[7] Distributed deep learning and inference without sharing raw data. https://splitlearning.github.io/