Evaluating Federated Learning Scenarios in a Tumor Classification Application

Rafaela C. Brum[1], **George Teodoro**[2], **Lúcia Drummond**[1], **Luciana Arantes**[3], **Maria Clicia Castro**[4], **Pierre Sens**[3]

[1]Fluminense Federal University – Niterói – Brazil

[2]Federal University of Minas Gerais – Belo Horizonte – Brazil

[3]Sorbonne Université, CNRS, INRIA, LIP6 – Paris – France

[4]State University of Rio de Janeiro – Rio de Janeiro – RJ – Brazil

{rafaelabrum, luciad}@id.uff.br, george@dcc.ufmg.br,

{luciana.arantes, pierre.sens}@lip6.fr, clicia@ime.uerj.br

Abstract. *Federated Learning is a new area of distributed Machine Learning (ML) that emerged to deal with data privacy concerns. In this approach, each client has access to a local and private dataset. They only exchange the model weights and updates. This paper presents a Federated Learning (FL) approach to a Tumor-Infiltrating Lymphocytes (TIL) application in a cloud computing environment. The results show that the FL approach outperformed the centralized one in all evaluated ML metrics. It also reduced the execution time although the financial cost has increased.*

1. Introduction

In recent years, the rise of data protection laws changed the way several applications handle their input data, asking for special permissions to manipulate data from different users. In the context of the Machine Learning (ML) area, when data is spread among different datasets and machines, processes of a distributed ML application exchange some data along the execution to train the model. However, in this new context, since data cannot be shared among different users, another approach to execute machine learning algorithms in distributed datasets appeared, where processes exchange only the model weights to obtain a global model, named Federated Learning (FL).

Biomedical applications, for example, which use patients' data, deal with confidential information. Several of these biomedical applications use a huge amount of data to predict a patient's diagnosis. In the case of image data, to extract some information from them, these applications, usually, execute a deep learning algorithm, like a Convolutional Neural Network (CNN). A CNN is composed of several layers. Most of them are either (i) convolutional layers, where the size of the image is reduced by applying a filter to it, or (ii) full layers (also called fully connected), containing many neurons, each one represented by a function. Training and extracting information from images, by using CNNs, require a huge amount of computational power, generally, obtained by using accelerators, like the Graphic Processing Units (GPUs).

Since several research groups do not have access to datacenters containing current GPUs, the cloud computing paradigm emerged as an interesting alternative. In clouds, the user only pays for what he/she uses (pay-as-you-go scheme), with no costs for hardware maintenance or energy consumption. Moreover, the user has access to the newest computational resources as soon as the provider makes them available to hire. In Amazon Web Services (AWS), for example, there are many types of virtual machines (VMs), also called instances, with different NVIDIA GPU architectures, from Kepler to Volta. Concerning the storage to keep the datasets, AWS also offers a service to store files with up to 5TB of size, called Amazon Simple Storage Service (Amazon S3) [1]. S3 offers privacy guarantees and lets the user grant different access permissions to each file.

In this paper, we present a Federated Learning approach to a Tumor-Infiltrating Lymphocytes (TIL) application that uses patient images to create TIL maps and predict the patient's survival rate. We implemented and evaluated this approach in the Amazon Web Services cloud provider. The experimental results showed that the proposed FL approach outperformed the centralized one in all evaluated ML metrics, when using the same number of training epochs, in a scenario with four clients, each one containing a private dataset of 1468 samples, 947 being used for training and 521 for testing. Also, it reduced the execution time by 55%, although has increased the financial cost by 46%.

2. Federated Learning concepts

Federated Learning (FL) was first introduced in [McMahan et al. 2017] when Google researchers presented the Federated Averaging (FedAvg) algorithm and evaluated it with three different datasets of image recognition and language modeling tasks. Later, they explained that this algorithm was evaluated in a real application, Google's Keyboard [2].

In the FL architecture, a central server communicates with all clients to manage the global model convergence. At each communication round, the server chooses the clients to participate in the training by sending them a start message. Each participating client trains its local model with its dataset on a fixed number of training epochs and sends the updated weights to the server. After receiving the message from the clients, the server aggregates the received weights to use them as the global model weights. This is the training phase of the communication round. The second phase is the testing one, where the server sends to the clients the aggregated weights of the training phase. The clients receive the weights, update their local models and test the model with their local test dataset.

3. Related work

Few papers found in the literature present a Federated Learning approach with cloud computing. Liu *et al.* [Liu et al. 2020] present a hierarchical FL architecture with mobile devices as clients. Their idea is to have edge servers aggregating the results of some clients and sending them to the cloud server. The central server aggregates the results of all edge servers. The authors present the algorithm using this architecture and the mathematical analysis of convergence. They only show simulation results to prove their convergence analysis. Fang *et al.* [Fang et al. 2020] propose an FL architecture in the cloud focusing on privacy. Their architecture consists of a central server on the cloud with clients that uses ElGamal encryption [ElGamal 1985] in all messages. They present how the server

[1]https://aws.amazon.com/s3/
[2]https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

and clients exchange their encryption keys as well as how the whole framework works. As the previous related work, they simulated the FL execution on a single local machine.

The only work we found that runs a Federated Learning approach in a real cloud environment implements two different machine learning (ML) methods to predict the risk of diseases related to tobacco and radon [Rajendran et al. 2021]. They implement a neural network and a logistic regression. The results compare a scenario with two clients in an asynchronous FL training where each client trains the model at a time.

To the best of our knowledge, most FL tools only support execution on a single machine, as the TensorFlow Federated [3] and the PySyft [Ryffel et al. 2018]. Both of them are FL libraries that simulate an FL environment based on a specific Machine Learning API: the former is based on the TensorFlow [Abadi et al. 2015] and the latter is based on the PyTorch [Paszke et al. 2019] one. Regarding the tools to execute FL in a real environment, Flower [Beutel et al. 2020] is presented as a framework that executes a Federated Learning approach in different devices and machines. The authors implemented the whole communication layer and also some of the first FL algorithms, including FedAvg. In this paper, we use the Flower framework as it is simple and allows us the use of any Machine Learning API underneath it.

4. A Use Case Application in Federated Learning approach

In this section, we present the developed Federated Learning approach applied for solving a Tumor-Infiltrating Lymphocytes (TIL) classification problem, as described in [Saltz et al. 2018]. The application solves the problem, receiving as input Whole-Slide Images (WSIs) and returning TIL maps. The TIL maps aim to obtain the spatial distribution and the density of TILs in each patient to help cancer treatment. Firstly, the application divides each WSI into patches of smaller sizes and classifies them as TIL-positive or TIL-negative by experts. The CNN uses the classified patches as a training dataset. These steps compose the application training phase. In the production phase, new WSIs are divided into patches and the CNN classifies these patches into TIL-positive or TIL-negative. With the TIL-positive patches, the application creates the TIL maps.

We focus only on the CNN training part implementing the VGG16 model [Simonyan and Zisserman 2015]. In this model, the used patch size is 224×224 and defines the patch dimension (height$\times$width). We also set the number of output classes as two, indicating positive if the patch contains a TIL or negative otherwise. The CNN uses two distinct datasets: one for training and another to test the model. Besides, the training dataset is randomly divided into a training and validate dataset. The evaluation of the proposed FL uses the centralized approach as a baseline.

In our FL approach, we implement the same VGG16 model in all clients and divide both datasets among them. Thus, each client has two distinct and private datasets: one for training and another for the test.

5. Experimental Results

Regarding the centralized approach, we selected 3793 patches as the training dataset and 2090 patches as the test dataset. They were obtained from the TCGA repository [4]. Both

[3]https://blog.tensorflow.org/2019/03/introducing-tensorflow-federated.html
[4]https://cancergenome.nih.gov/

datasets are unbalanced, with 10% of TIL-positive samples in the training dataset and 20% of positive samples in the test dataset. The CNN selected 10% of the training dataset to be the validation one. We executed 50 training epochs. We spent less than one dollar per hour executing the centralized approach in AWS, with a *g4dn.2xlarge* instance containing an Nvidia T4 Tensor Core GPU with 16GB of memory.

The Federated Learning approach divides the datasets among four clients homogeneously with only 10% of TIL-positive samples in each. Each client selected randomly 10% of the training dataset to be their validation dataset. We executed the clients in different *g4dn.2xlarge* instances and the server in a *t2.xlarge* one. We chose a cheaper instance for the server as it does not require a GPU. We considered four scenarios, varying the number of communication rounds within 50 training epochs in total. The first scenario has 25 communication rounds with 2 local training epochs each (25 comm. rounds). The second one has 10 communication rounds with 5 local epochs each (10 comm. rounds) while the third one has 5 communication rounds with 10 local epochs each (5 comm. rounds). Finally, the last scenario has only 2 communication rounds with 25 local training epochs each (2 comm. rounds).

Table 1 compares the results of the centralized approach and all scenarios. The columns show the model accuracy, precision, specificity, and total execution time and cost. The model accuracy is the percentage of well-predicted test samples, regardless of their true class. The precision is the percentage of true positives in all samples predicted as positive. The specificity shows the opposite, how many true negative samples were correctly predicted within all samples predicted as negative. The total time considers the moment from the request of the first VM up to the moment the last VM is terminated. The total cost includes the cost regarding all clients and the server for each FL scenario.

Tabela 1. Comparison among centralized approach and different number of communication rounds with 4 clients in Federated Learning

Scenario	Accuracy (%)	Precision (%)	Specificity (%)	Total exec. time	Exec. costs
Centralized	79.50	50.50	81.00	3:24:36	\$2.55
25 comm. rounds	91.29	-	91.29	2:31:13	\$7.92
10 comm. rounds	91.41	87.51	91.44	1:32:38	\$4.80
5 comm. rounds	91.20	49.31	91.63	1:09:13	\$3.53
2 comm. rounds	91.29	-	91.29	0:50:56	\$2.59

We can observe that the accuracy of all Federated Learning scenarios is better than the centralized one. As our dataset is unbalanced, when the FL approaches predict all samples as negative, they classify correctly most of the samples. For the second and third FL scenarios, we observe that the model yields a better precision and specificity than the centralized approach, which explains the better accuracy. We can observe that in the first and the last scenario the model did not predict any samples as positive. In the scenario with 25 communication rounds, each client trains for only two epochs per round, and, thus, the local models do not learn much from the data before communicating. So, at each communication round, the global model update does not result in significant improvements in the model. Consequently, in this first FL experiment, we obtained the worst results. When we compare the centralized approach with the best Federated Learning scenario (10 communication rounds with 5 training epochs each), we observe a decrease in the execution time by 55% with a corresponding cost increasing by 46%. Moreover, all metrics presented better results. Note that, although we used four GPU instances in FL, instead of only one, as in the centralized case, the costs increased only 46%. This happened because each client had a reduced dataset and spent much less time to finish. On the other hand, we did not reach a linear speedup due to the communication overhead.

6. Conclusion and future work

This work presented preliminary results of a Federated Learning approach to a Tumor-Infiltrating Lymphocytes CNN executed in Amazon Web Service (AWS). We executed four different scenarios varying the communication rounds number. Results show the FL approach leads to better results than the centralized one in a lower time, being a good method to deal with privacy concerns.

For future works, we will compare different CNN architectures into the FL approach and add more ML metrics to our results. Moreover, we plan to execute the FL approach using AWS cheaper instance types, the *spot* ones. These instances cost 70% less but can be revoked at any time. We intend to create a fault-tolerant framework to execute FL in AWS. Besides, we intend to consider the impact evaluation of using multiple clouds in the Federated Learning approach.

Referências

Abadi, M. et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Parcollet, T., and Lane, N. (2020). Flower: A friendly federated learning research framework. *ArXiv*, abs/2007.14390.

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472.

Fang, C., Guo, Y., Wang, N., and Ju, A. (2020). Highly efficient federated learning with strong privacy preservation in cloud computing. *Computers & Security*, 96:101889.

Liu, L., Zhang, J., Song, S., and Letaief, K. B. (2020). Client-edge-cloud hierarchical federated learning. In *2020-2020 IEEE International Conference on Communications*.

McMahan, B. et al. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proc. of 20th Int. Conf. on Artificial Intelligence and Statistics*.

Paszke, A. et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*.

Rajendran, S. et al. (2021). Cloud-based federated learning implementation across medical centers. *JCO Clinical Cancer Informatics*.

Ryffel, T. et al. (2018). A generic framework for privacy preserving deep learning. *ArXiv*, abs/1811.04017.

Saltz, J. et al. (2018). Spatial organization and molecular correlation of tumor-infiltrating lymphocytes using deep learning on pathology images. *Cell reports*.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *3rd Int. Conf. on Learning Representations, ICLR 2015*.