Abstract

Problem Statement:

We aim to develop a deep learning method that could extract COVID-19's graphical features in order to provide a clinical diagnosis ahead of the pathogenic test, thus saving critical time for disease control.

Solution Overview:

Dataset:

- 158 CT scan or X-ray images of 86 different patients infected with COVID-19, SARS and other pneumonia related diseases. (Dataset is being updated with new cases).
- Metadata of patients like age, sex, date of infection and clinical notes, etc.
- https://github.com/ieee8023/covid-chestxray-dataset

Proposed Solution:

Based on COVID-19 radiographical changes in CT images, we aimed to develop a deep learning method that could extract COVID-19's graphical features in order to provide a clinical diagnosis ahead of the pathogenic test, thus saving critical time for disease control.

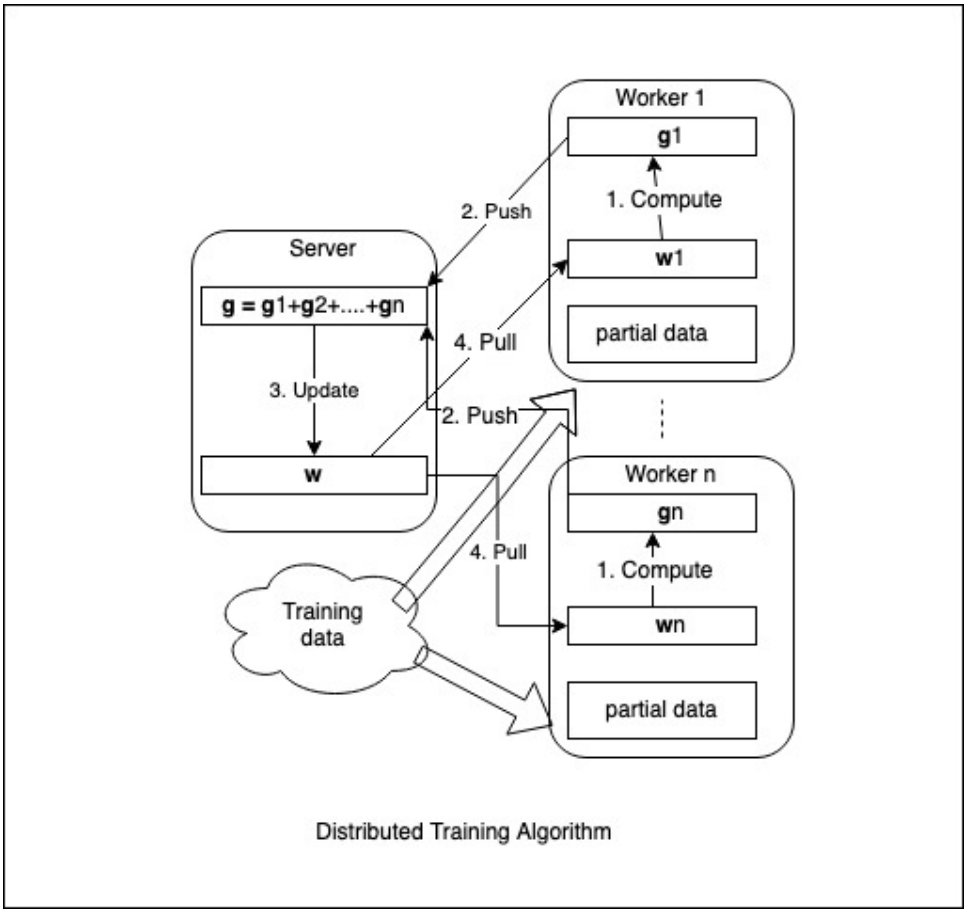
- Extraction of Region of interest images: Based on the signs of characteristic of pneumonia, ROI images are defined inflammatory lesions and can be extracted using computer vision techniques.
- Extract Features: The hallmarks of COVID-19 are bilateral distribution of patchy shadows and ground glass opacity. We will use CNN to identify unique features that might be difficult for visual recognition.
- Classification task: Classify images if patient is infected with COVID-19 viral pneumonia or not using ML model.

The dataset for this problem will keep growing with time. Therefore, it is important that training of the model is scalable. Hence, we will be implementing a distributed training algorithm.

Training the model:

We'll be distributing the task of optimizing the cost function. Computing gradients to help move the weights in right direction is computationally the heaviest task and hence we'll be parallelizing this task.

Data is divided into few partitions, with number of partitions equal to number of worker nodes. Each worker owns one independent partition and perform computation over that data and generate a set of parameter updates(gradients). All the nodes then synchronize their parameter states by network communication until they reach a consensus.



Distributed Training Algorithm

At a high level, the algorithm looks like the following on each *worker*:

1. On each worker, compute the gradient (partial derivative) on a subset of data
2. Push this partial gradient out to the server
3. Pull the new sets of weights from the server when server is ready

On the *server*:

4. Aggregate the gradients for all 'n' workers e.g. $g = \sum gi$

Server is the centralized unit with following functionalities:

Partitioning the data: We'll be partitioning the data using range partitioning.

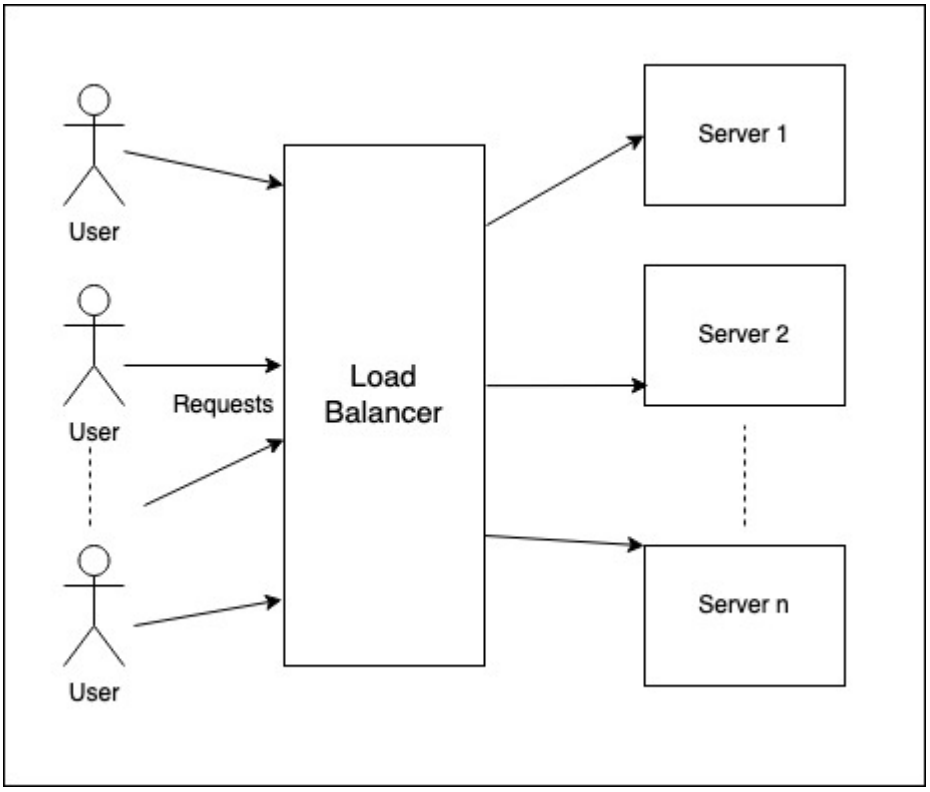
Monitoring the workers: using the heartbeat mechanism. If a worker goes down, then the data it was handling is assigned to other workers by the server.

Testing the model:

Scalability can be incorporated in the testing phase of the model where multiple user requests can be handled effectively by increasing number of instances serving model dynamically.

Load balancer will distribute user requests to multiple machines serving model instances and dynamically increase/decrease instances accordingly.

Test data will be stored locally at user's machine which will be sent to model running on remote machine and inference will be received at user's machine through network.



Team Members:

1. Niharika Khare (2018201002)
2. Pranjali Ingole (2018201029)
3. Ekta Agrawal (2018201043)
4. Kratika Kothari (2018201060)
5. Kshitij Paliwal (2018201063)
6. Anushka Wakankar (20171112)