

# Assignment 5 Report

## Timing Results Analysis

### Part 1 Timing Result

Average time per iteration: 1.9027502353374774 s

### Part 2 Timing Result

#### Node 0

Average time per iteration: 1.9757327055319762 s

#### Node 1

Average time per iteration: 1.9756835301717122 s

#### Node 2

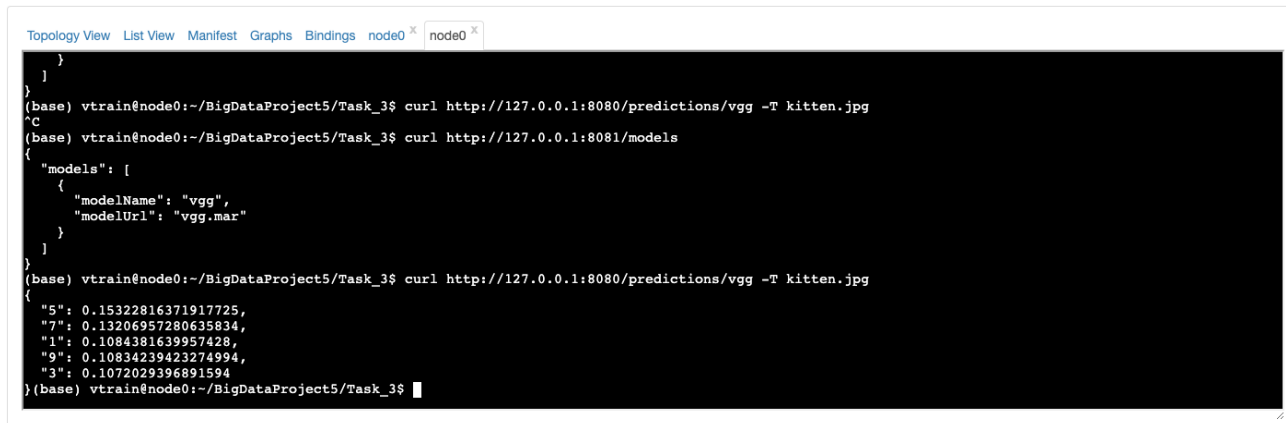
Average time per iteration: 1.9758784770965576 s

## Thoughts on Scalability of Distributed Machine Learning

In terms of average time per iteration, the distributed machine learning approach will tend to take longer due to the overhead of gradient reduction in order to update each weight in the network during backpropagation; to update the weights in the network, this requires an aggregation over all the generated gradients for a given weight that is potentially across multiple nodes, so communication costs will be high over the network, degrading performance time. This can be seen in above results where the average time per iteration for all the nodes was actually slightly more (about 0.07 seconds) than the time per iteration from Part 1. With that said, the benefits of this approach can be seen in terms of total time. Even though the time per iteration is a little more, since the iterations are being run in parallel, every node only runs the total number of iterations / number of nodes running. Due to this, even though the time per iteration is slightly more, the time per iteration is really divided by the number of nodes running. When

viewed from this angle, it is clear how even with a slightly higher average time per iteration, the distributed machine learning approach leads to a much faster overall runtime (being more scalable).

## Part 3 Screenshot



```
Topology View List View Manifest Graphs Bindings node0 node0
}
}
(base) vtrain@node0:~/BigDataProject5/Task_3$ curl http://127.0.0.1:8080/predictions/vgg -T kitten.jpg
^C
(base) vtrain@node0:~/BigDataProject5/Task_3$ curl http://127.0.0.1:8081/models
{
  "models": [
    {
      "modelName": "vgg",
      "modelUrl": "vgg.mar"
    }
  ]
}
(base) vtrain@node0:~/BigDataProject5/Task_3$ curl http://127.0.0.1:8080/predictions/vgg -T kitten.jpg
{
  "5": 0.15322816371917725,
  "7": 0.13206957280635834,
  "1": 0.1084381639957428,
  "9": 0.10834239423274994,
  "3": 0.1072029396891594
}
(base) vtrain@node0:~/BigDataProject5/Task_3$
```

## Specific Contributions of each Group Member

**Note: In general, all of the members worked together on all aspects and would really only work on the project when everyone was present. However, each member specialized in driving some aspect of the project.**

**Pranav:** Pranav helped in determining what logic should be written for the training loop and further tasks by looking through Pytorch API and relying on past knowledge from the neural networks class he took. This project required heavy usage of Pytorch to implement the training of a model normally, in a distributed manner, and on a server. Pranav led the coding aspect in general but also was assisted by his group members who also took the neural networks class. However, coding was still largely a group effort. Pranav also looked at documentation to meet the criteria of the assignment and worked towards making sure all of the files are well commented and fleshed out in terms of the READMEs and scripts.

**Viren:** Viren was in charge of helping provide some of the code for part 1 based on prior knowledge from the neural networks class that he had taken last semester. He also helped in running/serving the model in TorchServe and getting the prediction for the vgg\_model using TorchServe API. He also led in the report writing where he structured it and immensely helped in writing the comment regarding the scalability of distributed machine learning based on the timing results from Part 1 and 2. However, as mentioned at the very top of this section,

everything we did was a group effort. Everyone always showed up to all the meetups we had and we all evenly contributed.

Sameer: Sameer played a major role in ensuring the Python environment and necessary packages were set up properly along with the cluster so that the group was able to hit the ground running for the assignment. He was also heavily involved with integrating the Distributed Data Parallel framework into the code for Task 2's implementation and helped to iron out the logic behind partitioning of the training data using the distributed sampler. This section also required some initial debugging due to logic issues in our code, which Sameer also assisted greatly in addressing. However, this again was a group effort as mentioned before and virtually all work in code development was done as a collective effort with contributions from each member during group meetings.

## Other Implementation Details

Instead of using command

*python main.py --master-ip \$ip\_address\$ --num-nodes 3 --rank \$rank\$,*

we are using command

*python main.py --master\_ip \$ip\_address\$ --num\_nodes 3 --rank \$rank\$*

as we replaced hyphens in the first two arguments with underscores since we were getting errors when we tried to use the arguments with hyphens.

The implementation of our VGG Handler extended the existing code provided [here](#), also linked on the assignment page.