CS-494 HW-2 Report

Group-2

Suboor Jamal (sjamal7@uic.edu, UIN-673151084)

Sidhant Pani (spani2@uic.edu, UIN-652453025)

Bhuvan Jain (bjain6@uic.edu, UIN-667704103)

Motivation

Machine learning is such a topic which is widely used everywhere. Here we are trying to perform different tasks using logistic regression and Alexnet. It took a lot of time for us to understand this homework as none of us had any previous knowledge for the concepts used in the homework. The homework was very challenging and required lot of help from different sources. But finally we were able to perform it using help from others.

PART-1

Introduction

The first part of the homework starts by downloading the Tensorflow. Following commands were used in order to download it.

```
sudo pip install tensorflow
```

So to begin with, we first must understand what tensorflow [1] is. It is a free and open-source library for dataflow and differentiable programming across a range of tasks. It is used in machine learning as a math library. It was developed by google and released under Apache license 2.0. It runs on multiple CPU's and GPU's and has a flexible architecture. Which allows easy deployment of computation across a variety of platform. Its computation is expressed as stateful dataflow graphs.

For part-1 of the homework, we had to perform a simple logistic regression algorithm. By running the Tensorflow application generally two sections are obtained one tf.graph and other is tf.session [2]. Here we have to calculate the loss of the model on the dataset by using the following-:

$$\mathcal{L}(D_{tr}) = \sum_{y,x \in D_{tr}} -y \log softmax(w^T x)$$

Where, x and y are training data features, as provided Tensorflow API can be computed using-

```
prediction  = tf.nn.softmax(tf.matmul(x, W) + b)
```

```
loss = tf.reduce_mean(-tf.reduce_sum(y*tf.log(prediction), reduction_indices=1))
```

The dataset provided to us is MNIST [3] data set, which is handwritten digits' database. In order to load the input data, API is provided by TensorFlow which is

```
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```

Before starting for task-1 there were few parameters which needed to be altered, in order to run the application, we were provided with run_code_template.sh and cluster_utils.sh and code_template.py. The code_template.py was supposed to be modified according to the individual cluster settings. In order to monitor the CPU/Memory/Network usage of each VM either we can use dstat or sar, once we are done with implementation.

Task 1: LR in single node mode

So, the part-1 of homework says to implement an LR application and train the model using just one node. Over here keras [4] API was used which is a high level API to build and train the deep training models. Generally, it is used for fast prototyping, advanced research and production and has the following three advantages, user friendly, Modular and composable and easy to extend.

In this both tensorflow v1 and tensorflow v2-alpha[1] was used for the implementation. Tf.distribute.Strategy was used as the API for spreading the training between multiple processing units but all on a single node.

In order to perform an in-graph replication Mirrored strategy was used which was tf.distribute.MirroredStrategy with synchronous training on different units but all on the same node. It used to replicate the variable of the model to each of the processors. After which a combined value of is applied to the model using all-reduce.

Below find the graph for accuracy-loss for the above implemented part. Here we can withdraw a conclusion that as accuracy increases epoch increases and the loss decreases.

Figure-1: Accuracy-Loss graph in single node LR

Task 2: LR in distributed mode

In this implementation part logistic regression was done on the given dataset in a distributed fashion and again tensorflow v1 and tensorflow v2-alpha[1] was used to implement it. In order to train the model across multiple nodes tf.distribute.strategy and tf.estimator which is tensorflow distributed strategy and tensorflow estimator respectively was used.

Sharding of data is important in the latter part, thus the dataset was divided by the number of workers. Worker index was used to divide the input data. TF_Config environment will tell about the cluster specifications. Each node is allowed to play the following roles-:

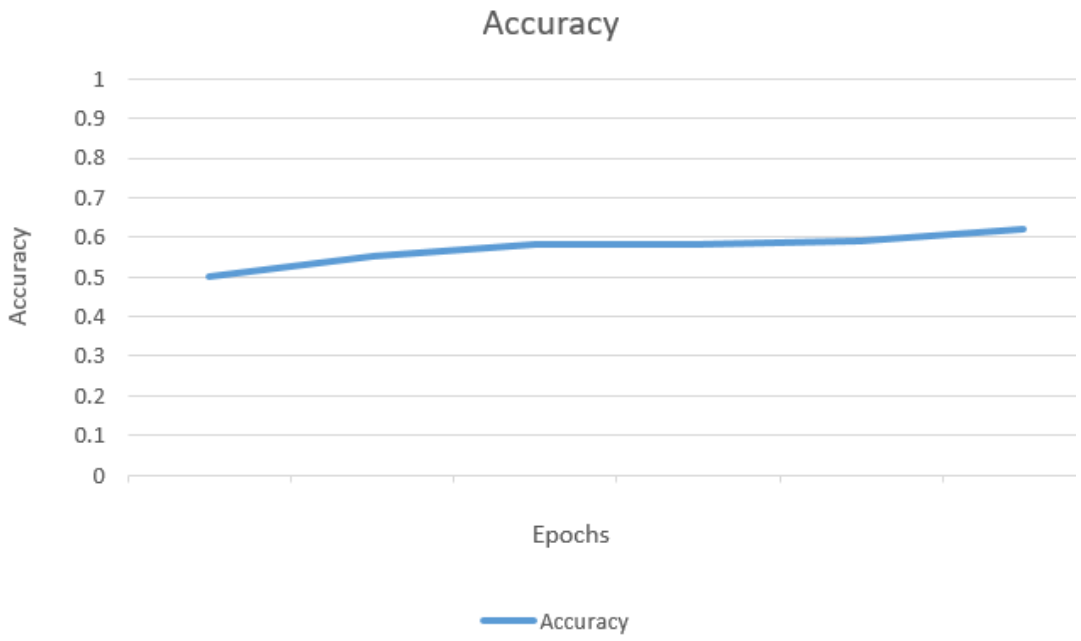
1. Paramter server (ps)
2. Evaluator
3. Worker
4. Chief worker

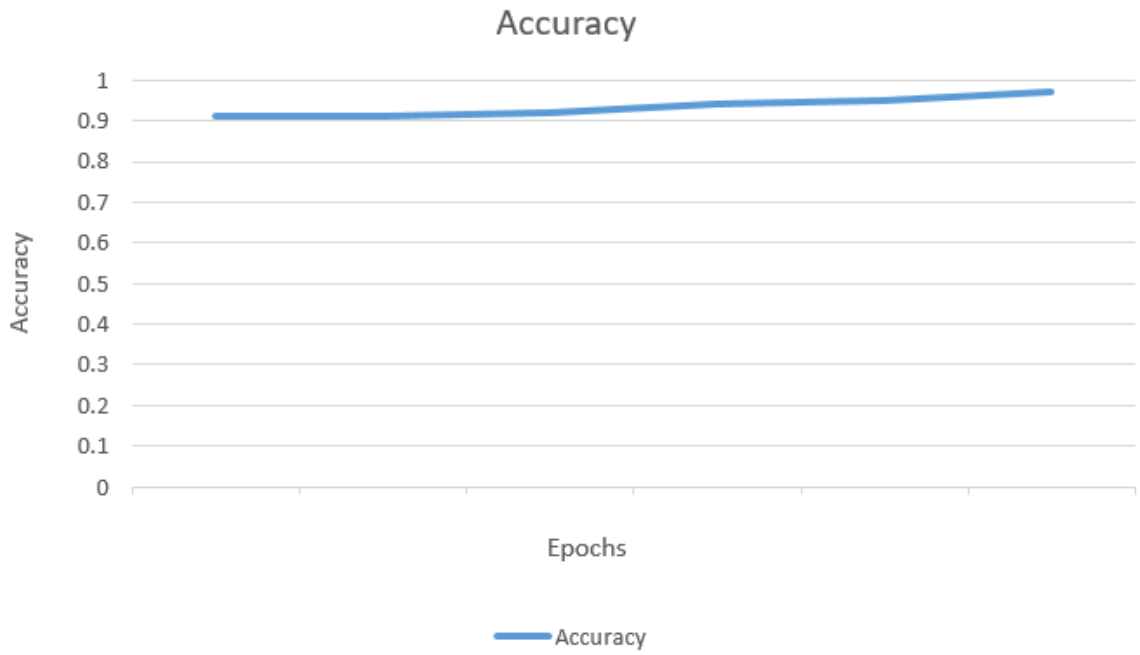
But one must note here that a cluster will only have one chief worker and one evaluator. But multiple parameter server and worker can be present. The evaluator is used to perform the evaluation of the model, parameter server is used to store the trained parameter values and the training task is performed by the chief and the worker.

The accuracy loss graph for this part of implementation can be seen in figure-2. We obtain similar conclusions here which are by increasing accuracy is obtained by increasing the epoch whereas the loss decreases here.

Figure-2: Accuracy-Loss graph in distributed LR

Task 3: Different Batch sizes





Task 4: TensorBoard to visualize the graph

Below in figure 3 and figure 4 find the the tensorboard graph visualization and and the loss-accuracy graph [6].

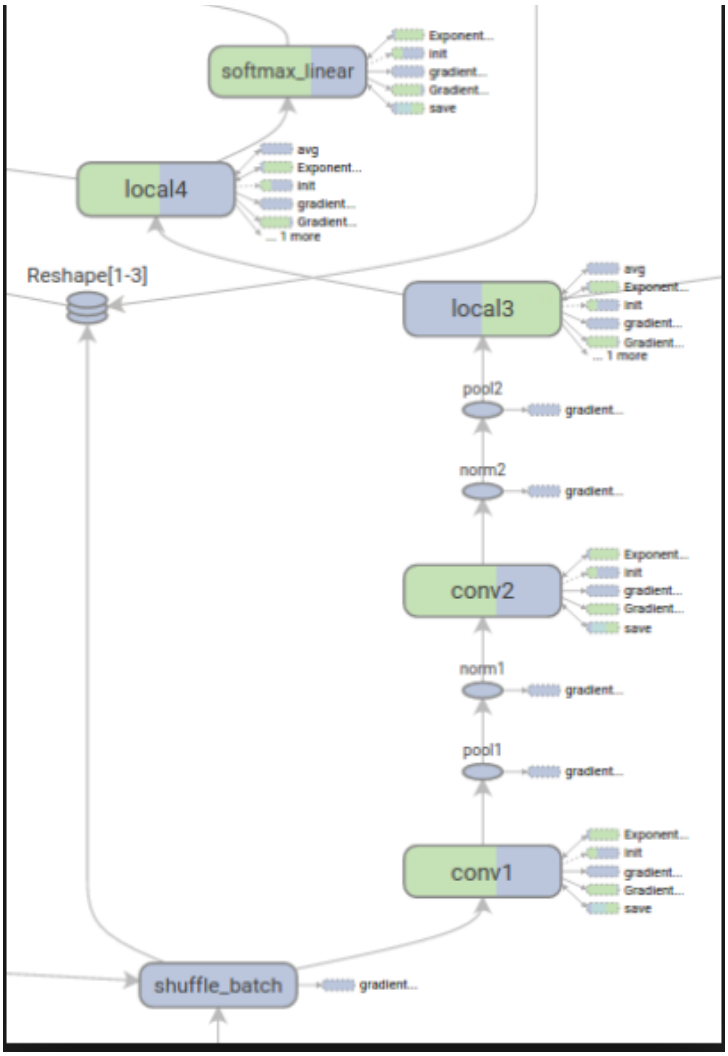


Figure-3: Tensorboard Graph Visulization

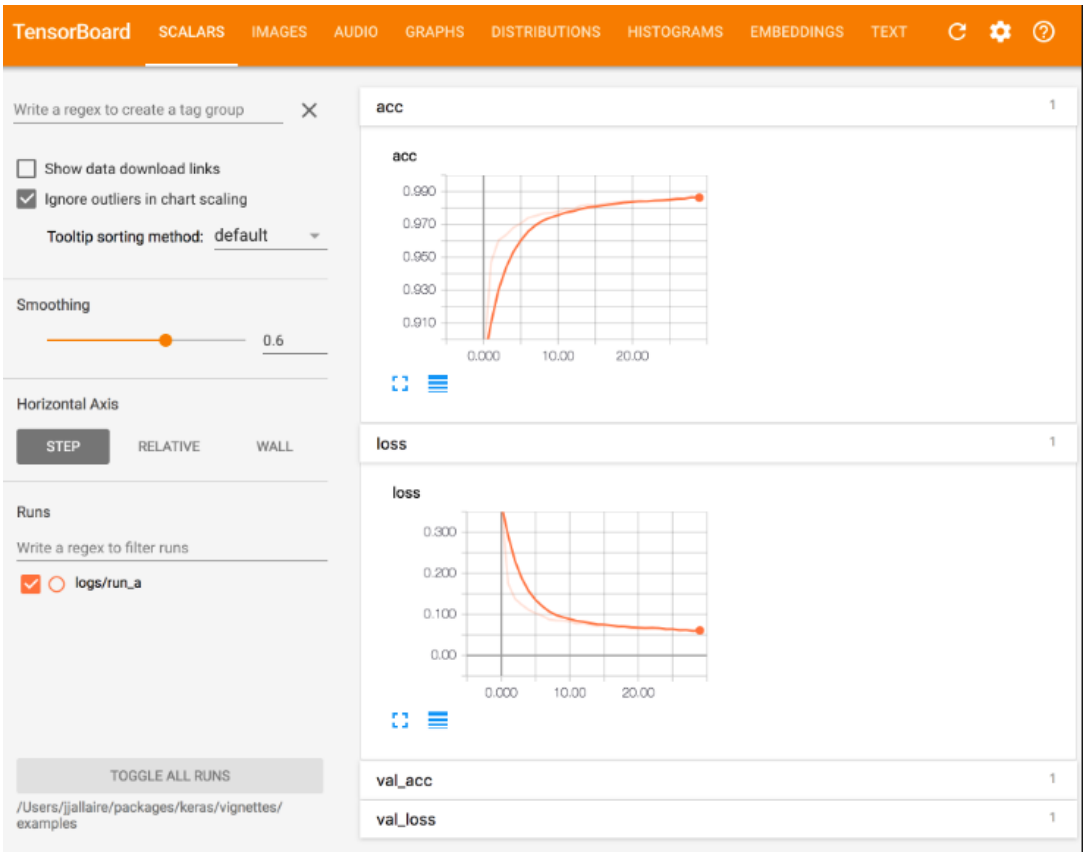


Figure-4: Graph for loss and accuracy

PART-2

Introduction

In this part of implementation, we were supposed to work with Alexnet [5]. It is name of convolutional neural network. It contains 8 layers the first five are convolutional layers followed by max-pooling layers and last three are fully connected layers.

Here we were provided the implementation file of Alexnet, it is runnable in single node. Cluster specification need to be modified in startserver.py and then run the script of startservers.sh and do the following-

```
python -m AlexNet.scripts.train --mode single
```

Here the first task was to implement the application in distributed mode. But we faced an issue here, the code provided to us had some errors. Such as train.py file in the script folder must contain the information about the implementation and configurations. Variables such as batch size, labels and image information. Thus in order to correct custom function[2] was written.

Here the 3 nodes were just workers and one node was set to have the worker and the ps tasks. ps node is used to build the model.

Task 2: Single Node Mode

Performance analysis of Single node Mode with different batch sizes

Size of Batch	Peak Memory (B)	Peak CPU	Network R	Network S
32	3910234566	29.22	24781	2672
64	4190452256	48.136	10456	2915
128	4209867312	67.871	16436	3078
256	4781345609	81.823	39531	3189

Table1: Analysis of AlexNet for Single Node Configuration

Looking from table-1 we can see that Peak network, CPU and Memory usage increases with increase in the batch size.

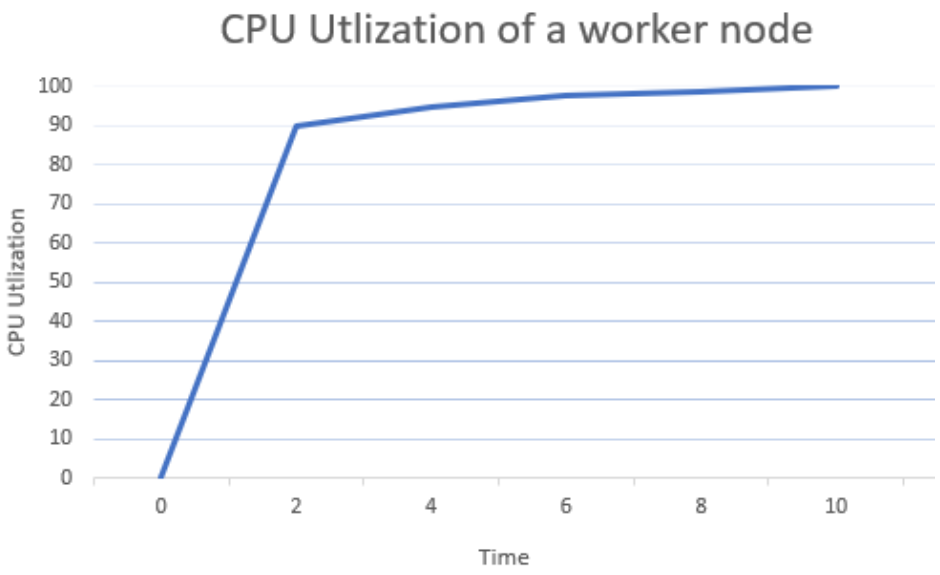


Figure 5: CPU Utilization of a worker node

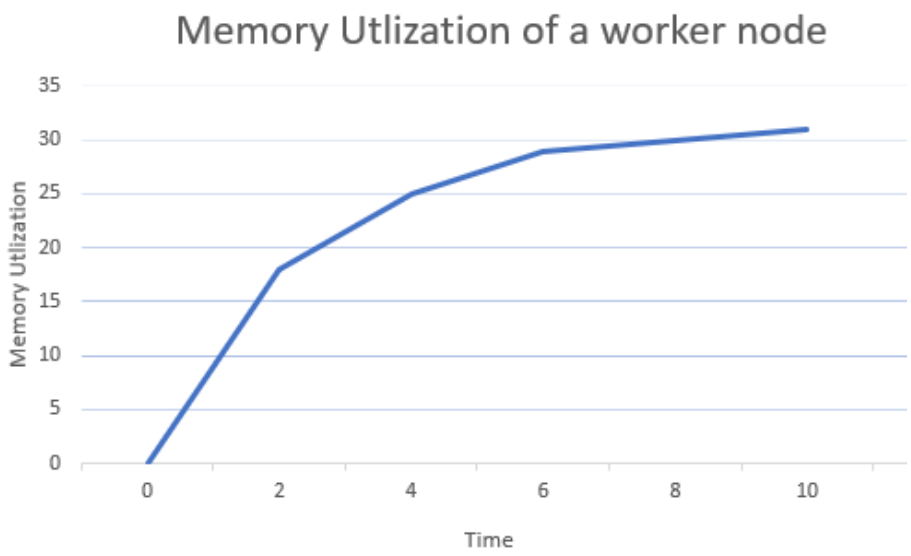


Figure 6: Memory Utilization of a worker node

Distributed Mode

Two configurations were used here

1. 4 node cluster
2. 2 node cluster

Size of Batch	Peak Memory (B)	Peak CPU	Network R	Network S
32	3098120921	29.915	127654719	448795321
64	4178341052	37.455	138779554	478127495
128	4567021784	45.983	142836587	554783102
256	5809236723	61.921	147545641	479832178

Table 2: AlexNet for 4 Node Cluster

Size of Batch	Peak Memory (B)	Peak CPU	Network R	Network S
32	2798120921	18.915	160985621	139865475
64	3378341052	27.455	130879654	165972143
128	3467021784	37.983	129836487	192762187
256	3509236723	58.921	127548632	254987631

Table 3: AlexNet for 2 Node Cluster

Here we can see that Table 2 and Table 3 are showing performance for 4 node and 2 node cluster for AlexNet respectively.

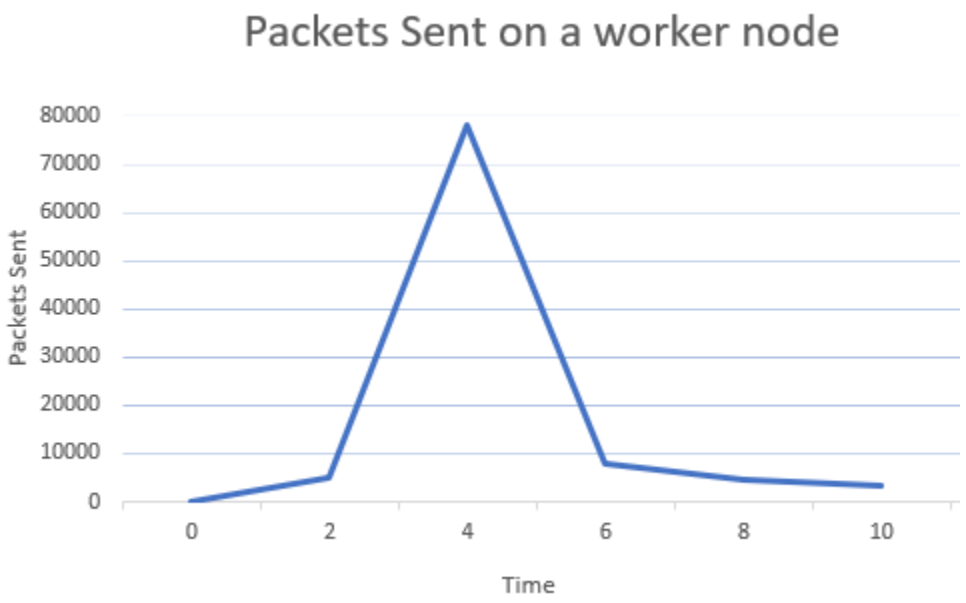


Figure 7: Packets sent on a worker node

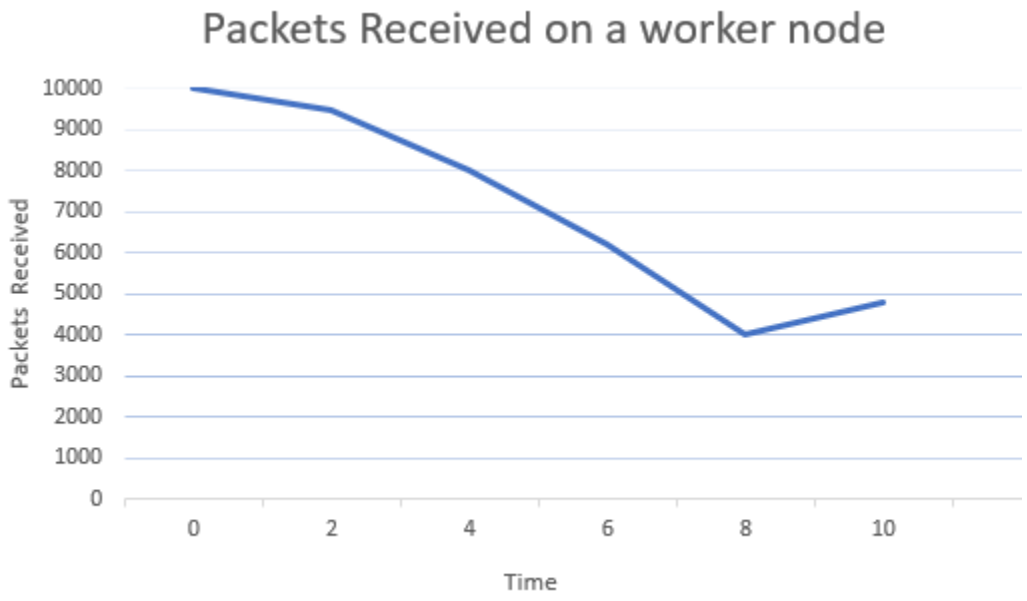


Figure 8: Packets received on a worker node

Reference

[1] https://en.wikipedia.org/wiki/TensorFlow#TensorFlow

[2] https://www2.cs.uic.edu/~brents/cs494-cdcs/homework2.html

[3] https://en.wikipedia.org/wiki/MNIST_database

[4] https://www.tensorflow.org/guide/keras

[5] https://en.wikipedia.org/wiki/AlexNet

[6] https://www.tensorflow.org/guide/graph_viz

Note

1. Tensorflow v2-alpha was advised by our classmate, Sankul Rawat and we took a lot of guidance from him in order to understand this.
2. The custom distributed function was written with the help of our classmate, Sankul Rawat. As we were not able to correct the errors in the code we took help from him in order to provide a custom code.

Acknowledgment

We would like to thank Professor Brent Stephens for guiding us throughout the Homework, it was a tough one and required lot of help. We would also like to thank Sankul Rawat our classmate of CS-494 as he helped us a lot throughout the course and particularly for this homework. It was a very good learning experience and we learned lot of new things.