Distributed Inference With Straggler Mitigation

Lolla Sai Krishna

A Thesis Submitted to

Indian Institute of Technology Hyderabad

In Partial Fulfillment of the Requirements for

The Degree of Master of Technology

Department of Electrical Engineering

June 2019

Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.

L.Saikrishna

(Signature)

Lolla Sai Krishna

(Lolla Sai Krishna)

EE17AMTECH11003

(Roll No.)

Approval Sheet

This Thesis entitled Distributed Inference With Straggler Mitigation by Lolla Sai Krishna is approved for the degree of Master of Technology from IIT Hyderabad

28/6/19

(Dr.Subrahmanya Sastry Challa) Examiner
External Examiner
Dept. of Mathematics
IITH

28/6/19

(Dr.G.V.V.Sharma) Examiner
Internal Examiner
Dept. of Electrical Engineering
IITH

28/06/19

(Dr.Lakshmi Prasad Natarajan) Adviser
Dept. of Electrical Engineering
IITH

28/06/19

(Dr.Aditya Siripuram) Co-Adviser
Dept. of Electrical Engineering
IITH

28/6/19

(Dr.G.V.V.Sharma) Chairman
Dept. of Electrical Engineering
IITH

Acknowledgements

I sincerely thank Dr.Lakshmi Prasad Natarajan & Dr.Aditya Siripuram for their guidance & support throughout out my thesis. I would also like to extend thanks to my Lab mate Mr.Suman Ghosh for his valuble suggestions & his assistance in writing my thesis.A special thanks to my friend Mr.Dinesh Jain who helped me in the techincal aspects & also the programming part in my thesis.Lastly i would thank my parents for their support throughout my life.

Dedication

I dedicated this thesis to my parents.

Abstract

In today's world machine learning has major applications in a wide variety of tasks such as image classification,object detection and natural language processing.Machine learning models are trained and deployed in prediction based cloud services which are mostly prediction serving systems.These systems take input requests from users & return predictions by performing inference on trained model. These services use a distributed architecture for serving user requests which consist of many nodes which are inter connected.These nodes face a number of unavailability's such as temporary slowdowns and failures.Nodes facing temporary unavailability are known as stragglers.These nodes delay the entire process of computation.

The objective of this thesis work is to design a framework for inference in a distributed setup which is robust to stragglers.The distributed setup is trained in such away that it classifies the image with good accuracy even in the presence of straggling nodes during inference.

Distributed setup consists of many neural networks in parallel along with a master neural network also known as decoder which collects the prediction vetor from all nodes.The image to be classified is partitioned into as many number of parts as there are nodes and is given as input to each node.The final predictions are taken from decoder.

Two neural network architectures are considered one being base-MLP model and the other one being being CNN model while implementing the distributed setup. The setup is trained for various possible cases of straggler scenarios.During training phase decoder and nodes are learned by back propagating the error & updating weights through gradient descent algorithm.

During inference the setup is tested by varying the number of stragglers in the test set after training the setup for stragglers.

The distributed setup classifies the image with good accuracies during during inference even in the presence of stragglers in the input as it gets trained for different possible scenarios of stragglers during training phase.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Distributed Computation

In today's world machine learning serves as a backbone for a wide variety of tasks such as image classification,speech recognition ,autonomous driving in cars etc. Distributed compute architecture is required to run large scale machine learning algorithms.These machine learning algorithms involve multiplication of matrices of massive sizes which cannot be run on single machine.These operations involving massive computations are parallelized across multiple nodes(workers).The real time data that is used for experimentation using machine learning algorithms is very large in size .To withstand against massive computational load and requirement of memory for processing huge datasets which can't be handled by a single machine , distributed computing infrastructures have been proposed with the objective of speeding up in a processing a big task which is distributed across several nodes(workers) running in parallel.

Each and every node in a large scale distributed system will be assigned a subtask.So each of these nodes(workers) will be running in parallel with a central node called master node which collects the results from all the working nodes to complete the task.

In distributed computing scenario, the master node has to wait for all worker nodes in order to complete the task.One of the major bottleneck in distributed computing is straggler nodes or unresponsive nodes which may be due to shared resources,system failures.These nodes increase the latency or execution of task as the master node waits to collect results from all nodes including the straggler nodes.

1.2 Related Work

Straggler mitigation techniques have been addressed in [3] using error correction codes by adding redundancy.The naive way of adding redundancy is to replicate the task given to a working node where multiple copies of same task will be given to all working nodes so that the whole distributed system can tolerate upto $n-1$ stragglers where n is the number of straggling nodes.This is equivalent to adding redundancy using repetition code from coding theory perspective.By replicating tasks and collecting the outputs from fast responsive worker nodes can significantly decrease the latency of the entire computation or task.

Another error correcting code known as erasure code or MDS(Maximum Distance Separable) code [4] is also used to introduce redundancy .An erasure code introduces redundancy by encoding k message symbols into a codeword of length n where (n>k).It has the property that any k coded symbols are enough to recover the original message. It means in a distributed computing scenario the results from any k sub tasks are sufficient for the master node to do further processing. In the above discussion straggling mitigation techniques have been discussed for linear computations like matrix-matrix multiplication & matrix-vector multiplication.

Non-linear computations like image classification using different neural network models in a distributed scenario have been addressed in [2] where redundancy is introduced by a parity neural network . Here a distributed scenario consisting many pre-trained neural networks(nodes) are present in parallel.Different images from the same dataset are given as input to the nodes. These images are combined and given to an encoder neural network to produce parity image which is the form of redundancy.The predictions are all neural networks including encoder are concatenated and given to master neural network known as decoder. .Final output predictions are taken from this decoder.Encoder and decoder neural networks are learned for straggler cases by back propagation in training phase.Stragglers are assumed by replacing the prediction of any node by a row vector of zeros.The main drawback is as the number of working nodes increase the accuracy of predictions decrease as only one parity image needs to contain the information about the remaining images.Moreover this can handle only one straggler at a time.

Collage-CNN models are discussed in [5] where redundancy is introduced in the form of a collage.Multi image classification is done at one shot by giving this collage as input to the CNN model. In a distributed system of N nodes, N-1 nodes are fed with N-1 single input images which are different. All single input images are lowered in resolution and combined to forming a collage which is given as input to last node.The decoder reconstructs the missing prediction vector based on the prediction vector obtained from collage CNN model. The main drawback is this system can tolerate only one straggler at a time. As the number of single images per collage increase, number of objects to be detected by the Collage-CNN is also higher & the resolution of each single image in collage also gets decreased.This is the main cause for decrease in accuracy.

1.3 Overview of Thesis Work

- Inference refers to the special case of Image Classification.
- Distributed setup consists of multiple neural networks in parallel with a master neural network also known as decoder to which predictions from neural networks are given as input.Final predictions are taken from decoder.
- Stragglers are mimicked by replacing the pixel intensity values present in the input image to -1.
- The distributed setup is trained for different cases of stragglers.
- Then inference is done by testing the distributed setup by varying the number of straggling nodes in the test data set.

Chapter 2

Background Theory

2.1 Artificial Neural Network

Neural network is a entity that does processing of a signal similar to the visual cortex of human brain which contains millions of neurons. Neural networks consists of many layers with each layer containing some set of artificial neurons or nodes for the processing the signal[1]. The output of each layer is computed by applying non-linearity or linearity to weighted sum of inputs.The connection between nodes are called as weights.Input signal will pass through different layers starting from input layer to output layer with a non-linear activation function applied on top of each layer.

2.2 Convolutional Neural Network

Convolutional Neural Networks(CNN) are a special class of neural networks which are used for classification of images.CNN's have a reported success on MNSIT dataset. CNN's generally consist of 3 different types of layers :Convolutional Layer ,Pooling Layers ,Fully Connected Layers.

Convolution Layer

Convolutional Layer consists of several filters.Each filter will be slid across the entire image and element wise dot product is performed for the values in the kernel with the corresponding pixel values in that portion of image on which kernel is placed.The result of element-wise product is obtained as a single value which is a scalar.Then bias values are added. This is followed by applying a non-linearity as an activation function which is generally RELU or Sigmoid.Convolutional Layer are known to extract the edge features of an image.

Pooling Layer

Pooling Layers are used to reduce the spacial dimensions of an image.Two types of pooling layers are preset one being max pooling layer & the other one being average pooling layer. Max pooling layer produces an output corresponding to the maximum value of all neuron units present in the prior layer.Average pooling takes the average of pixel values present in a specific layer .

Fully Connected Layer

In fully connected layer each neuron is connected to every other neuron by a set of weights.Generally fully connected layers are present at the end to obtain the values that are used classify the image.The disadvantage with this layer is that number of trainable parameters increases by increasing these layers.

Training Process

Neural networks are trained by learning.Learning happens through updating the trainable parameters.Training involves the selecting the model architecture , forward propagation & backward propagation operations through the network, updating the training parameters by back propagating the loss through the network.Weight parameters are updated iteratively updation of weights by calculating the gradient with respect to corresponding inputs followed by gradient descent algorithm.

Chapter 3

Inference using MLP Architecture

Each & every node follows Multi Layer Perceptron Architecture.Each node consists of 1 Input Layer & 1 Output Layer with sigmoid activation function . Inference is done by testing the distributed setup with different scenarios of straggling nodes.

3.1 Image Partitioning

Here the dataset considered is MNIST.Each image in the dataset is grey scale and is of size 28x28 .This image is flatten down to a 784-length vector and is then zero padded at the end.This is partitioned as shown is Fig.3.1 into as many of number of parts as there are no.of nodes present in the setup excluding decoder.The zero padding is done at the beginning or at the end . Adding zeros doesn't change the pixel information present in the image. The parts of input which are obtained after partitioning are given as inputs for different nodes present in the network.

This operation can also be seen as shifting & downsampling .This is done to reduce the inherent redundancy present in the image.So even if there no inherent redundancy present in the image the system should classify the image with correct label.(Accuracy of prediction should be high.)

If the image is just cut(sliced) into three parts successively , the image can be still predicted because of the inherent redundancy.

3.2 Architecture

Distributed Setup consists of many neural networks in parallel along with decoder at the end.Each image in the training dataset is partitioned as shown in Fig.3.1 and parts obtained after partitioning are given as input to each node present in the distributed setup.The predictions from each node are concatenated to form a big vector.This concatenated vector is given as input to the decoder.The final predictions for the labels are taken from decoder .Two cases of distributed setup one with $N = 10$ nodes & the other with $N = 50$ nodes are considered for experimentation which will be explained in the following sections.

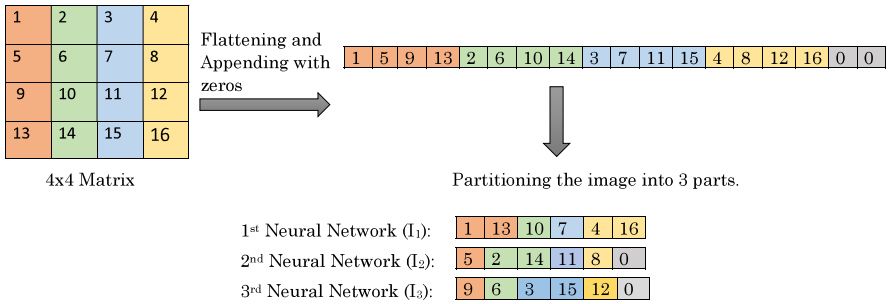

Figure 3.1: Example for partitioning a simple 2-D Image

Architecture of Distributed Setup for 10 & 50 nodes

The architecture for a distributed setup consisting of $N = 10$ nodes & $N = 50$ nodes is shown in Fig.3.2 & Fig.3.3 respectively.Each node(neural network) along with the master neural network(decoder) in the setup follows a simple architecture consisting of an input layer & an output layer on top of which sigmoid activation is applied with no hidden layer in between the input and output layers.Architecture of each node and master neural network for both cases is shown in Table 3.1 & Table 3.2 respectively.

Table 3.1: Architecture of Each Node

InputShape	No Of Nodes	No Of Layers	Layer Type	Layer Size	ActivationFunction	Parameters
(1x80)	10	1	FullyConnected	(80x10)	Sigmoid	800
(1x17)	50	1	FullyConnected	(17x4)	Sigmoid	68

Table 3.2: Architecture of Master Neural Network(Decoder)

InputShape	No Of Nodes	No of Layers	Layer Type	Layer Size	ActivationFunction	Parameters
(1x101)	10	1	FullyConnected	(101x10)	Sigmoid	1010
(1x201)	50	1	FullyConnected	(201x10)	Sigmoid	2010

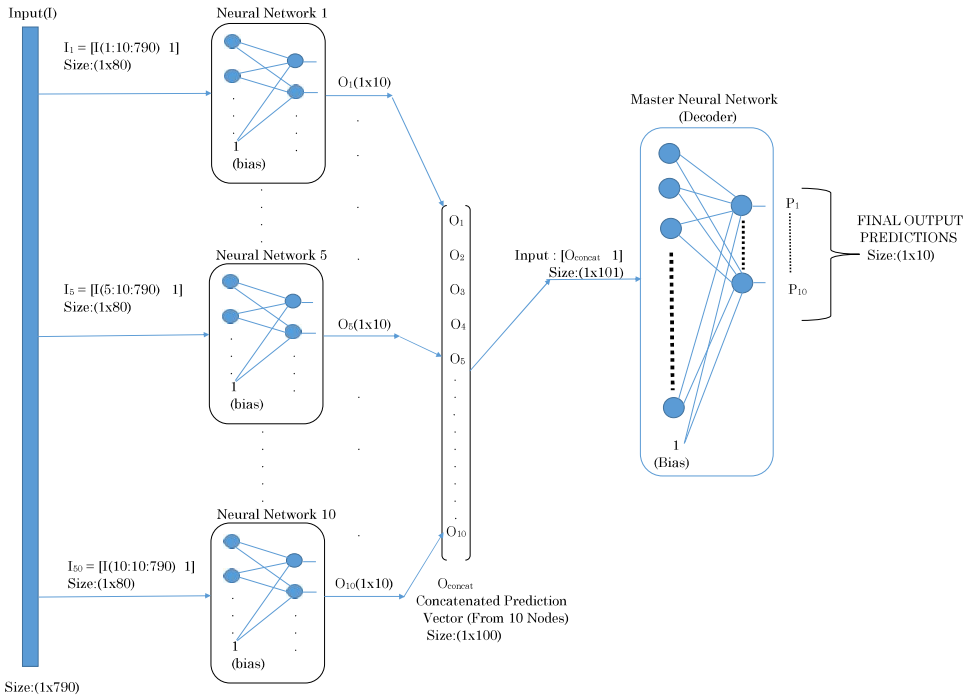

Figure 3.2: Architecture For 10 Nodes

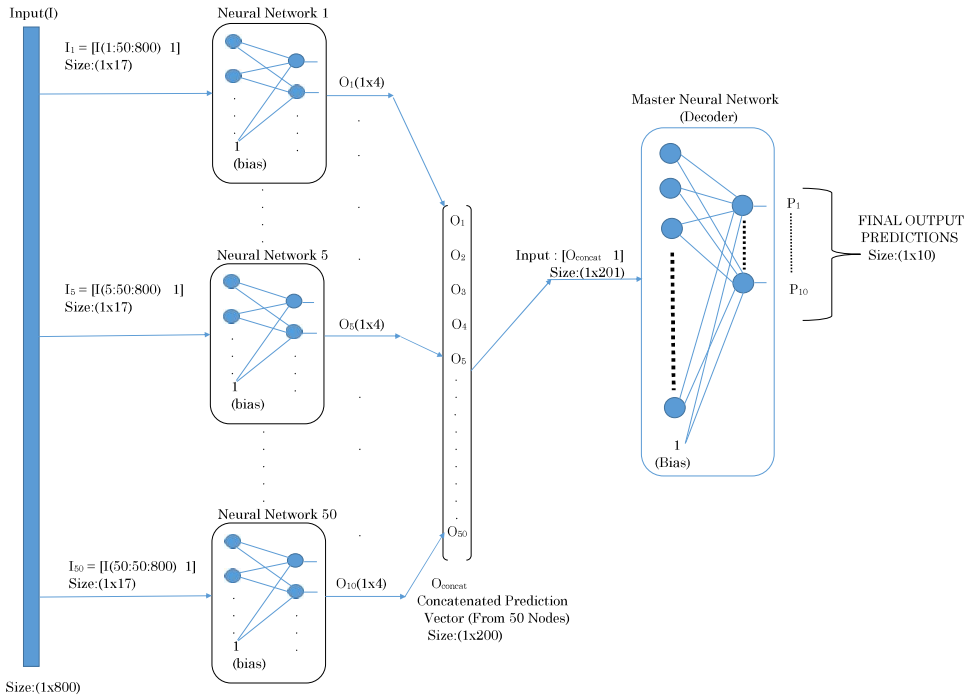

Figure 3.3: Architecture For 50 Nodes

3.3 Training Process

Training process involves training the each and every node present in the network along with decoder.In order to make the distributed setup robust to stragglers the network is trained for various possibilities of stragglers.

The images present in the MNIST dataset are having pixel intensities which vary from 0 to 255 with 0 being black and 255 being white.Values in between these correspond to different shades present in the gray color. These pixel values in the image are scaled to range 0-1 before giving as an input to the neural network.

Stragglers are introduced in the training dataset by replacing the pixel values to -1 which is out of the range 0-1 so that the neural network cannot classify the image correctly .Stragglers are considered by randomly choosing the input partitions and replacing all the pixel values present in chosen parts completely with -1. The error is back propagated through the master neural network to all the nodes(neural networks) by updating all the weights corresponding each node as well as weights at the decoder.

Decoding Process

Weights corresponding to each node & decoder are obtained after training the setup for stragglers. For example in a system consisting of 3 nodes consider the case of one node being a straggler,three possible cases will be there.Distributed setup is trained for all three cases & weights corresponding

to each node is obtained. During inference if first node is straggler(first input partition replaced with -1) first input partition will get multiplied with weights corresponding to first node(obtained from training) to obtained prediction vector for first node.The prediction vector from the other nodes is concatenated with this first prediction vector & is given as input to decoder.

Process of training the distributed setup is explained as follows:

- Partitioning of image into parts.
- Select a random number between 0 & n(where n = number of stragglers) with uniform probability.
- Randomly select the parts which corresponds to random number obtained from previous step of total input partitions.
- Replace the pixel values in randomly choose parts with -1.
- Forward propagate through all nodes to decoder.
- Calculate between true label & final prediction.
- Back propagate loss through decoder to all nodes.Update weights of all nodes & decoder by considering batches using batch gradient descent.
- This process is repeated for many of epochs in order to train the setup for more number of straggler scenarios.

The algorithm for training is implemented in MATLAB R-2018a

3.3.1 Algorithm For Training & Testing Process

The algorithm for training & testing the distributed setup is explained in this section.

- Here the distributed setup is trained for 0-n stragglers.

Algorithm 1 Training Algorithm

Input:No of Nodes(q),Epochs,BatchSize,Prediction Vector Length in every node(m)
I = 784 (No of Pixel Values), Input vector length per each node = $\lceil \frac{784}{q} \rceil$
Output: Weights
Initialize weights
while $i < Epochs$ **do**
    $i \leftarrow i + 1$
    N=$60,000$ (No. of training samples)
    for $j = 1$ *to* N **do**
        Append the required no of zeros at the end to the input.
        $s = randi([0, n], 1, 1)$ (where n = number of stragglers)
        if $s = 0$ **then**
            Partition the input.
        else
            $s1 = s$
            Partition the input .
            $s2 = randperm(q, s1)$
            Replace all the elements(pixel values) present in those inputs pointed out by $s2$ with -1.
        end if
        Forward Propagate through all nodes(neural networks) including decoder.
        Calculate the error & back propagate it through the respective parts of input.
        Update the weights corresponding to q neural networks & also the weights corresponding to decoder.
        end for
        end while

Algorithm 2 Algorithm For Calculating Test Accuracy

Input:No of Nodes(q),Iterations,Weights obtained from training algorithm
Output : Accuracy
I = 784 (No of Pixel Values), Input vector length per each node = $\lceil \frac{784}{q} \rceil$
while $i < iterations$ **do**
    $i \leftarrow i + 1$
    N2=$10,000$ (Number of samples in the test dataset)
    for $i = 1$ *to* N2 **do**
        Append the required no of zeros at the end to the input vector.
        Partition the input.
        $s = randperm(q, h)$ (where n = No.of Stragglers in the test dataset).
        Replace pixel values in the partitioned input parts pointed out by s vector with -1.(Introducing Stragglers in the test dataset).
        Forward Propagate through all nodes(neural networks) including master neural network to calculate the predictions.
        Count = Count +1 (if predicted vector matches label vector)
Accuracy = $\frac{Count}{iterations * \text{N2}}$

- Test Accuracy is calculated for h stragglers in the test dataset.

3.3.2 Training Parameters

Experimentation is carried out for the case of 10 & 50 nodes.Loss function considered is Mean-SquaredError.Training process uses mini batches of 20 samples for above mentioned both cases respectively.The whole distributed setup is trained with a learning rate of 1.5 .The weights for all the fully connected layers in each node as well as the weights in the master neural network are initialized based on Normal distribution $N(0, 0.00001)$.All the biases are initialized to 1.The number of epochs for which the setup is trained was 100 .No of iterations during testing were 100.

The total number of trainable parameters corresponding to weights are $800 \times 10 + 1010 = 9010$ & $68 \times 50 + 2010 = 5410$ for 10 & 50 nodes respectively.

3.4 Results & Discussion

The test accuracy of the distributed setup which is trained for different possible scenarios of stragglers versus number of straggling nodes in the test dataset is considered for evaluation.

3.4.1 For the case of 10 nodes

A. Distributed Setup trained for 0 stragglers

Here stragglers are not introduced into the training dataset.

Table 3.3: Trained for 0 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	96.08
1	75.89
2	73.48
3	49.77
4	39.86
5	35.09
6	22.04
7	18.97
8	15.64
9	12.27

B. Distributed Setup trained for 0-1 stragglers

Here the setup will be trained for 0 as well as 1 stragglers during training.Every sample in the training set after being subjected to partitioning will consist of 1 randomly chosen part as stragglers(1 randomly chosen input partition will be replaced with -1) or no stragglers.

Table 3.4: Trained for 0-1 stragglers

Stragglers in Test Set	Test Accuracy(%)
0	96.34
1	95.82
2	95.22
3	94.23
4	92.85
5	90.26
6	85.82
7	76.19
8	57.33
9	32.38

C. Distributed Setup Trained for 0-2 stragglers

Here the setup will be trained for 0 ,1 & 2 stragglers. Every sample in the training set after being subjected to partitioning will consist of 2 randomly chosen parts as stragglers(i.e, two randomly chosen input partitions will be replaced with -1) or 1 randomly chosen part as stragglers(1 randomly chosen input partitions will be replaced with -1) or no stragglers.

Table 3.5: Trained for 0-2 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	96.09
1	95.99
2	95.53
3	94.71
4	93.49
5	92.09
6	88.04
7	83.24
8	69.73
9	43.37

D. All the cases of training the distributed setup from 0-3 stragglers to 0-9 stragglers

Here i present the results of test accuracy for different cases of stragglers starting from training from 0-3 stragglers to 0-9 stragglers.

Table 3.6: Trained for 0-3 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	96.30
1	96.11
2	95.48
3	94.81
4	94.06
5	92.61
6	90.09
7	84.81
8	74.16
9	45.97

Table 3.7: Trained for 0-4 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	96.26
1	95.96
2	95.47
3	95.06
4	94.34
5	93.00
6	90.89
7	86.32
8	76.32
9	52.79

Table 3.8: Trained for 0-5 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	96.09
1	95.82
2	95.64
3	95.22
4	94.64
5	93.38
6	92.08
7	88.49
8	78.72
9	58.88

Table 3.9: Trained for 0-6 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	95.97
1	95.66
2	95.41
3	95.04
4	94.49
5	93.84
6	92.37
7	89.54
8	80.54
9	62.47

Table 3.10: Trained for 0-7 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	95.85
1	95.22
2	95.01
3	94.72
4	94.30
5	93.61
6	92.53
7	90.51
8	85.85
9	64.91

Table 3.11: Trained for 0-8 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	95.25
1	95.05
2	94.87
3	94.59
4	94.20
5	93.65
6	92.73
7	90.60
8	87.08
9	73.04

Table 3.12: Trained for 0-9 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	94.93
1	94.72
2	94.50
3	94.23
4	93.63
5	93.46
6	92.46
7	91.02
8	87.82
9	78.86

E. Plot of Test Accuracy

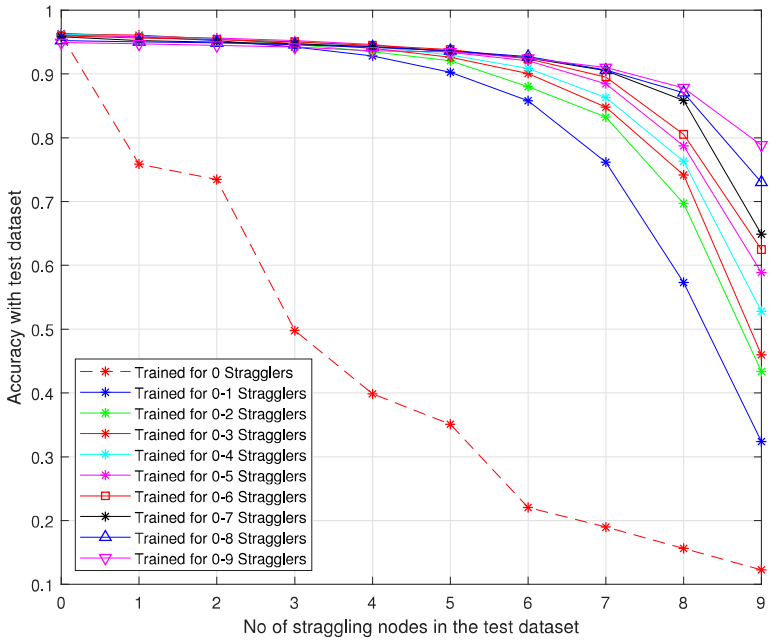

Figure 3.4: Accuracy with test dataset for 10 nodes

From the Fig.3.4 it is observed that there is an increasing trend in test accuracy when the distributed setup is trained for more number of stragglers.For example when the setup is trained for 0-6 stragglers ,the validation accuracy for 8 stragglers in the test dataset is 80.54% but if the same setup is trained for 0-7 stragglers the validation accuracy increases to 85.85%.We can also observe from Fig3.4 that upto 3 stragglers in the test dataset the validation accuracy for all cases of training remains almost the same excluding the case of training for 0 stragglers.

F. Overall Computation Time(in seconds)

$$T_{\text{Overall}} = T_{\text{Node}} + T_{\text{Decoder}} \tag{3.1}$$

$$T_{\text{Node}} = \max_{i \in \{1,2,\ldots,q\}} T_i$$

$$\text{Number of flops at each node } = m \times [2\lceil I/q \rceil + 1]$$

$$T_i = \alpha \times m \times [2\lceil I/q \rceil + 1] \quad \forall i \in \{1, 2, \ldots, q\}$$

$$\text{Number of flops at decoder } = P \times [2qm + 1]$$

$$T_{\text{Node}} = \alpha \times m \times [2\lceil I/q \rceil + 1]$$

$$T_{\text{Decoder}} = \alpha \times P \times [2qm + 1]$$

where α = Time taken per 1 flop in sec/flop
$T_i = T_{\text{Overall}}$=Time taken at each node
m = Prediction vector length in each node(m)
q = Number of nodes
P = Final Prediction Vector
$T_{overall}$ = Overall time taken for image classification. Hence from eq. 3.1 we obtain

$$T_{\text{overall}} = \alpha \times [m \times [2\lceil I/q \rceil + 1] + P \times [2qm + 1]] \tag{3.2}$$

Overall computational time obtained is a function of q & m Here the overall computational time in this case is 3600 seconds.

- Here were are considering combinatorial model where all nodes are assumed to have same computational capacity(power).
- Time taken by each node will be same for all nodes.
- Time taken at each node is directly proportional to the number of computations(flops) at each node.
- Without loss of generality we assume $\alpha = 1$.(Time taken per flop is 1 sec)

3.4.2 For the case of 50 nodes

Here i present the results of test accuracy versus the no of stragglers in the test set for 50 nodes.

Table 3.13: Trained for 0 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	93.84
10	62.28
20	33.73
30	22.28
40	12.21
49	9.95

Table 3.14: Trained for 0-10 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	93.57
10	92.23
20	89.28
30	79.88
40	50.86
49	16.90

Table 3.15: Trained for 0-20 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	93.65
10	92.50
20	90.04
30	85.04
40	63.96
49	11.89

Table 3.16: Trained for 0-30 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	92.94
10	92.39
20	90.73
30	86.80
40	69.77
49	15.01

Table 3.17: Trained for 0-40 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	92.50
10	91.91
20	90.57
30	87.67
40	75.26
49	19.05

Table 3.18: Trained for 0-49 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	92.05
10	91.42
20	90.08
30	87.53
40	78.19
49	26.61

A. Plot of Test Accuracy versus No of Stragglers in the test dataset

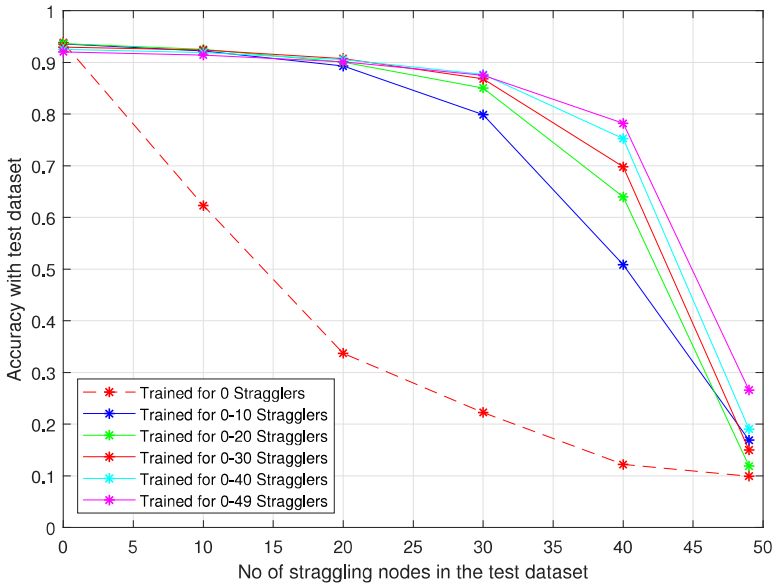

Figure 3.5: Accuracy with test dataset for 50 nodes

From Fig.3.5 it is observed that the distributed setup can tolerate upto 20 stragglers in the test dataset.

B. Overall Computation Time(in seconds)

The overall time taken for image classification is calculated according to 3.1 .Any system should account for scalability.So the total time taken for image classification should be same for both 10 & 50 cases.Overall time taken is made independent of number of number of nodes present in the setup by varying the input size of each node & prediction vector(m) in each node.

3.4.3 For the case of 10 nodes with reduction in the size of training set

Here the distributed setup is trained by randomly picking 10000 instead of using the entire training set. Reduction in size is done to see the how the test accuracy varies with training dataset size.

Table 3.19: Trained for 0 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	93.36
2	76.33
4	48.98
6	26.85
8	16.74
9	15.18

Table 3.20: Trained for 0-2 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	93.74
2	93.00
4	90.03
6	85.68
8	67.70
9	44.35

Table 3.21: Trained for 0-4 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	93.88
2	93.21
4	91.87
6	88.64
8	73.37
9	42.64

Table 3.22: Trained for 0-6 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	93.90
2	92.66
4	91.94
6	89.77
8	79.86
9	58.37

Table 3.23: Trained for 0-8 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	93.21
2	92.73
4	91.96
6	90.07
8	84.01
9	68.31

Table 3.24: Trained for 0-9 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	92.62
2	92.27
4	91.72
6	90.32
8	84.91
9	75.33

A. Plot Of Test Accuracy

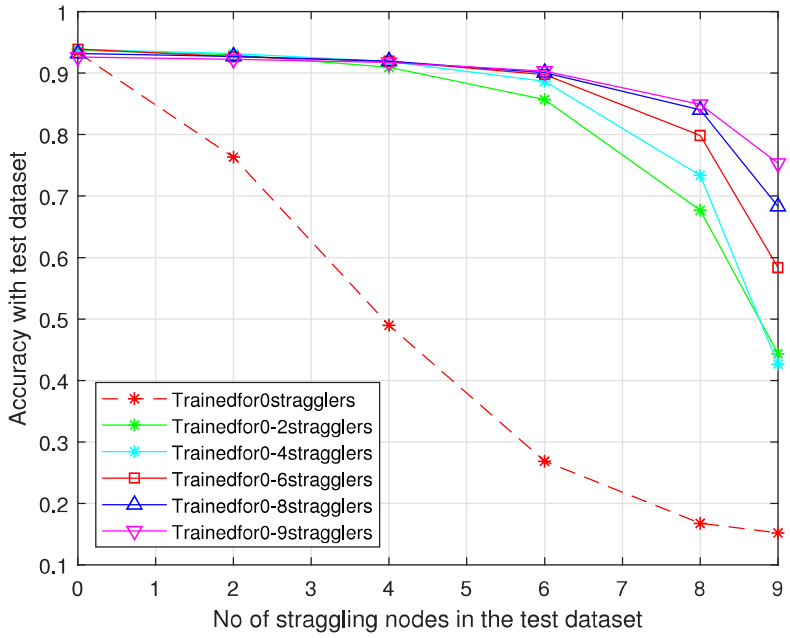

Figure 3.6: Accuracy with test set by reducing training set size

3.5 Conclusion

From the above results we can conclude that when the distributed setup gets trained for more and more stragglers in the training phase it becomes robust to more stragglers during inference. An increasing trend in the test accuracy values is observed as the the setup is trained for more stragglers.

Overall time is made independent of number of nodes present in the setup by varying the prediction vector length in every node and size of the input given to each node.

As the number of nodes increase , the prediction vector length is reduced in each node so that the computational complexity at the decoder deceases at the cost of decrease in the prediction accuracy of label.

The distributed setup can tolerate arbitrary number of stragglers. But the accuracy decreases(graceful degradation) with increase in the number of stragglers.

Even with reduction in the size of training dataset the distributed setup is able to classify the image with test accuracy above 90% upto 4 stragglers in the test dataset excluding the case of trained for 0 stragglers as shown in Fig.3.6.The accuracy decreases by 2-3%(graceful degradation) compared to their counter parts using the entire training set.

3.6 Comparison

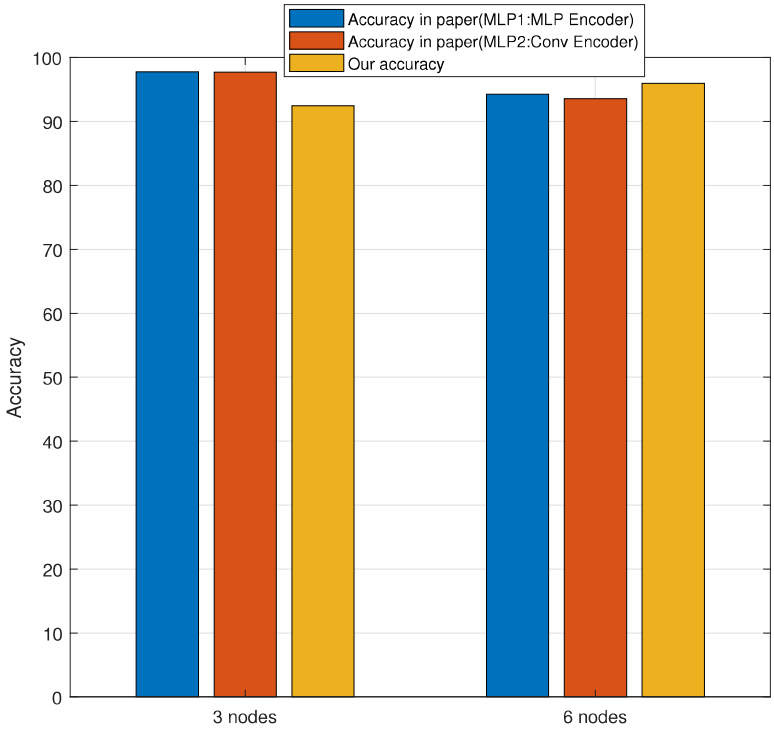

Figure 3.7: Comparison with [2]

- Here we compare our accuracy values with the ones with the ones obtained in [2].
- Comparison is with respect to MNIST dataset.
- Two cases are considered one for 3 & another one for 6 nodes.Each node follows MLP architecture.
- Loss function considered is MSE(Mean Squared Error).
- As the number of nodes increase,our distributed setup gives better accuracy compared to its counterpart in [2].
- For both cases the distributed setup is trained for exactly one of the node being a straggler.Also during inference the is distributed setup is tested for exactly one out of all nodes being a straggler.
- The main drawback with [2] is as the number of working nodes increase the accuracy during inference decreases .

Chapter 4

Inference using CNN architecture

Here image classification is done in a distributed setup consisting of many nodes which are CNN'S followed by a decoder which consists of simple MLP-architecture.The reason for choosing CNN architecture is they are known to perform well by giving better classification results by extracting complex features present in an image compared to MLP. The dataset here refers to only MNIST & Fashion MNIST.

4.1 Image Partitioning

Every image in the training set is resized appropriately using bi cubic interpolation and is then partitioned. This is done to avoid more number of zero vectors as input to each CNN.

4.2 Experimentation on MNIST Dataset

4.2.1 Architecture

The architecture consists of several CNN's in parallel followed by a decder which is a simple MLP.

For 10 nodes

Here i present the architecture of each CNN & decoder .Each of the 10 CNN's follow the same architecture.As shown in Fig.4.1 & Table 4.1 each CNN has 2 convolutional layers, 2 max pooling layers & a fully connected layer.Decoder is a simple MLP consisting of 2 fully connected Layers with softmax activation function applied on last layer.

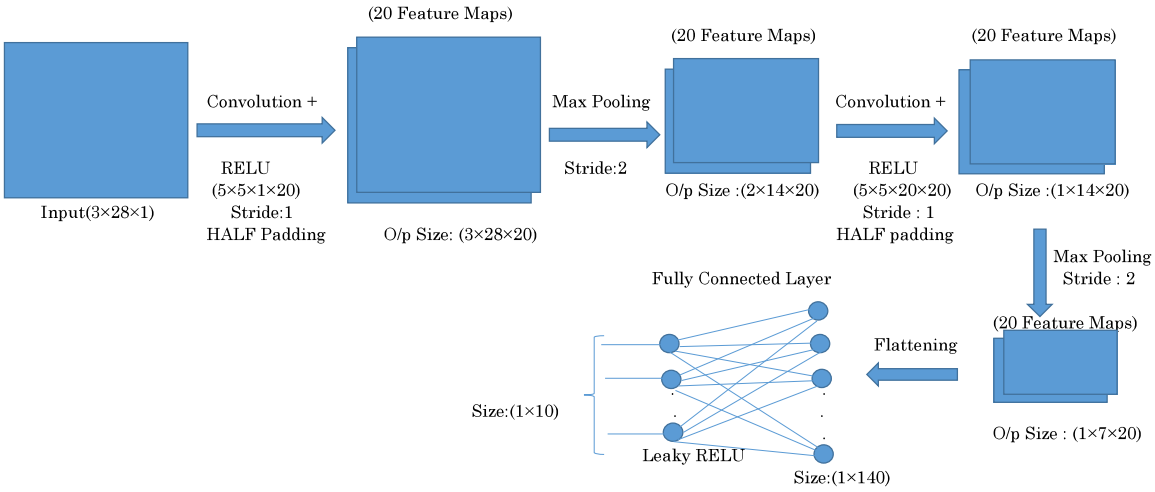

Figure 4.1: Architectural View of each CNN for 10 nodes

Table 4.1: Architecture Of Each CNN model

Layer	Output Size	Parameters	Activation
Input	(None,None,3,28,1)	-	-
1st Conv	(None,3,28,20)	520	LeakyRelu
Maxpooling 2D	(None,2,14,20)	-	-
2nd Conv	(None,2,14,20)	10020	LeakyRelu
Maxpooling 2D	(None,1,7,20)	-	-
Flatten	(None,140)	0	-
FullyConnected	(None,10)	1410	LeakyRelu

Table 4.2: Architecture Of Decoder

Layer	Output Size	Parameters	Activation
Input	(None,100)	0	-
FullyConnected	(None,500)	50500	LeakyRelu
FullyConnected	(None,10)	510	Softmax

For 50 nodes

Architectural Overview is shown in Fig.4.2

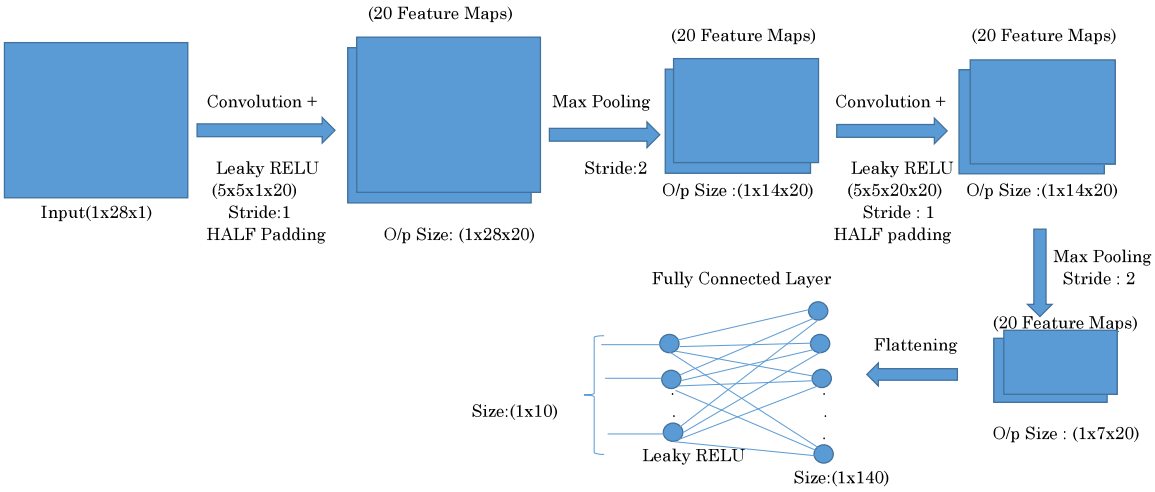

Figure 4.2: Architectural View of each CNN for 50 nodes

Table 4.3: Architecture Of Each CNN model

Layer	Output Size	Parameters	Activation
Input	(None,1,28,1)	0	-
1st Conv2D	(None,1,28,20)	520	LeakyRelu
Maxpooling 2D	(None,1,14,20)	0	-
2nd Conv2D	(None,1,14,20)	10020	LeakyRelu
Maxpooling 2D	(None,1,7,20)	0	-
Flatten	(None,140)	0	-
FullyConnected	(None,10)	1410	LeakyRelu

Table 4.4: Architecture Of Decoder

Layer	Output Size	Parameters	Activation
Input	(None,450)	0	-
FullyConnected	(None,500)	225500	LeakyRelu
FullyConnected	(None,10)	510	Softmax

4.3 Process of training

Training process is explained in the following steps

- Firstly the size of training set is increased.
- Each image is resized to size 30×28 using bicubic interpolation & is partitioned.

- A random number is chosen between 0 & n (where n = number of stragglers) with probability.
- Input partitions are chosen randomly depending upon the value of random number obtained in the previous step.
- Replace the pixel values in chosen input partitions with -1.
- Forward Propagate through all nodes to the decoder for obtaining final predictions.
- Loss is calculated between true labels & prediction vector and is back propagated through decoder to all nodes.
- Weights are updated at all nodes including the decoder.

4.4 Training Parameters

I present the training hyper parameters used for the training process in detail.

- Batch Size : 200
- Learning Rate : 0.002
- Training DataSet Size : 1,20,000
- Optimizer : Adam
- Loss Function : Cross Entropy

The size of kernels & biases for different layers and their distributions are shown in Table 4.5

Table 4.5: Weights and their distributions for convolutional layers

Item	Layer	Size	Distribution
Weights	1^{st}Conv	(5,5,1,20)	Truncated Normal with mean = 0 & standard deviation = 0.07
Weights	2^{nd}Conv	(5,5,20,20)	Truncated Normal with mean = 0 & standard deviation = 0.07
Biases	1^{st}Conv	(1,20)	Truncated Normal with mean = 0 & standard deviation = 0.05
Biases	2^{nd}Conv	(1,20)	Truncated Normal with mean = 0 & standard deviation = 0.05

- Truncated Normal Distributions are drawn from range[-2,2].
- Weights for fully connected layers in each CNN as well as decoder are initialized are initialized according to truncated normal distribution with mean 0 & standard deviation 0.003
- Biases for fully connected layers in each CNN as well as decoder are initialized are initialized according to truncated normal distribution with mean 0 & standard deviation 0.006
- Number of epochs for training as well as testing :100

4.5 Results & Discussion

During inference Test accuracy is measured by varying the number of stragglers present in the test dataset after training the setup for different cases of stragglers.

4.5.1 For 10 nodes

Table 4.6: Trained for 0 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	97.39
1	61.87
2	31.81
3	25.78
4	19.47
5	18.90
6	15.53
7	12.56
8	10.40
9	10.22

Table 4.7: Trained for 0-1 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	97.79
1	96.95
2	94.72
3	91.49
4	84.93
5	81.26
6	69.19
7	54.47
8	43.14
9	23.04

Table 4.8: Trained for 0-2 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	97.57
1	96.81
2	95.71
3	93.86
4	91.46
5	85.04
6	78.43
7	63.95
8	48.55
9	31.99

Table 4.9: Trained for 0-3 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	97.71
1	96.74
2	95.56
3	94.05
4	91.32
5	87.19
6	82.30
7	71.59
8	57.31
9	34.34

Table 4.10: Trained for 0-4 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	97.42
1	96.39
2	96.09
3	94.34
4	92.54
5	88.59
6	83.63
7	74.59
8	62.07
9	41.94

Table 4.11: Trained for 0-5 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	96.92
1	97.08
2	95.84
3	94.77
4	92.62
5	89.52
6	85.08
7	76.33
8	63.98
9	44.47

Table 4.12: Trained for 0-6 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	97.49
1	96.50
2	95.88
3	94.23
4	92.82
5	89.66
6	85.1
7	76.52
8	65.52
9	46.45

Table 4.13: Trained for 0-7 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	97.10
1	96.06
2	95.34
3	93.70
4	92.15
5	89.66
6	84.81
7	77.59
8	67.18
9	45.89

Table 4.14: Trained for 0-8 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	97.13
1	96.18
2	95.17
3	93.55
4	92.18
5	88.91
6	84.79
7	77.36
8	68.16
9	48.26

Table 4.15: Trained for 0-9 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	96.70
1	95.99
2	94.60
3	93.75
4	91.33
5	88.30
6	84.06
7	77.42
8	68.23
9	49.68

A. Plot of Test Accuracy

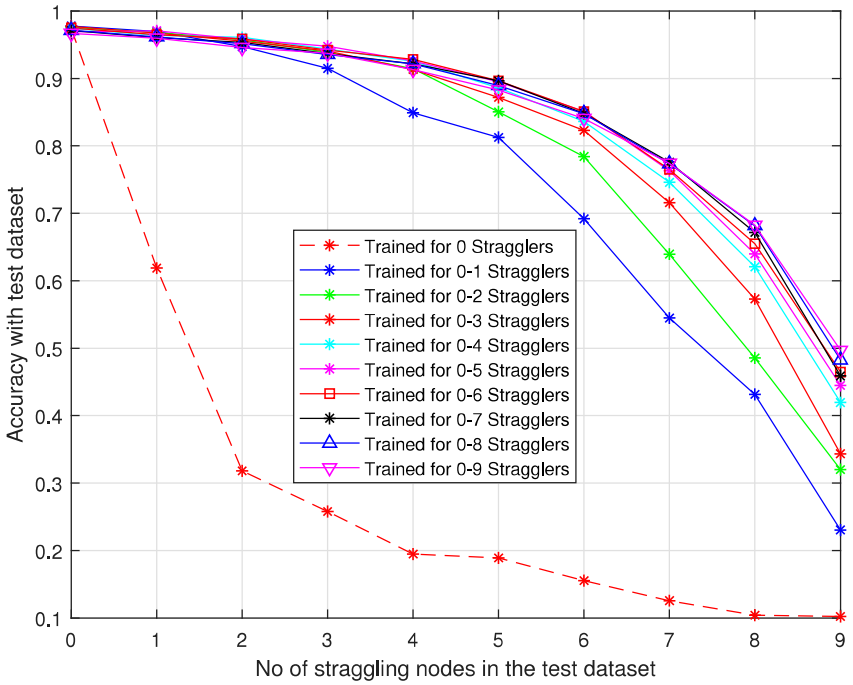

Figure 4.3: Plot of Test Accuracy for 10 nodes

B.Overall Time Taken(in seconds)

$$T_{\text{Overall}} = T_{\text{Node}} + T_{\text{Decoder}} \tag{4.1}$$

The time taken at each node is proportional to the number of flops(computational complexity) at each node.

$$T_{\text{Node}} \;=\; \alpha \times (\text{Flops at each node}) \qquad\qquad T_{\text{Decoder}} \;=\; \alpha \times (\text{Flops at decoder})$$

where α = Time taken per 1 flop in sec/flop.

Computation Time of Convolution & Pooling Layers

Let the original image size be (m, n, p).Then after resizing the size of image which is to be fed into each CNN becomes $(\lceil m/q \rceil, n, p)$ where q is number of CNN's present in the distributed setup.

- When this input is convolved with a kernel of size $(n.n, p, f_n)$ the time taken at the convolution layer will be

$$T_{\text{conv}} = \alpha \times \lceil m/q \rceil \times n \times f_n \times [2n^2 p - 1] \tag{4.2}$$

  where f_c = number of channels present in the image , f_n = number of kernels of size(n, n, p).Here the padding is taken as same which is also known as half padding where padding factor = $\lfloor n/2 \rfloor$.

- When this input is fed through a maxpooling layer with stride s the time taken by max pooling layer will be

$$T_{\text{maxpool}} = \alpha \times s^2 \times \lceil m/s \rceil \times \lceil n/s \rceil \times p \tag{4.3}$$

- Without loss of generality we assume $\alpha = 1$.(Time taken per flop is 1 sec)

- Time taken by each node T_{node} is calculated as follows:

$$\begin{aligned}
T_{\text{1stConv}} &= 3 \times 28 \times 20 \times [2(5^2) - 1] = 82320 \\
T_{\text{1stMaxpool}} &= 4 \times 2 \times 14 \times 20 = 2240 \\
T_{\text{2ndConv}} &= 2 \times 14 \times 20 \times [2(5^2)(20) - 1] = 559440 \\
T_{\text{2ndMaxpool}} &= 4 \times 1 \times 20 = 560 \\
T_{\text{Fullyconnected}} &= 10 \times [140 + 140 - 1] = 2790
\end{aligned}$$

$$T_{\text{Node}} = T_{\text{1stConv}} + T_{\text{1stmaxpool}} + T_{\text{2ndConv}} + T_{\text{2ndmaxpool}} + T_{\text{fullyconnected}} = 647350 \tag{4.4}$$

- Time taken by decoder T_{Decoder} is calculated as follows :

$$\begin{aligned}
T_{\text{1stFullyConneceted}} &= 500 \times [2(100) - 1] = 99500 \\
T_{\text{2ndFullyConnected}} &= 10 \times [2(500) - 1] = 9990
\end{aligned}$$

$$T_{\text{Decoder}} = T_{\text{1stFullyConnected}} + T_{\text{2ndFullyConnected}} = 109490$$

$$T_{\text{Overall}} = T_{\text{Node}} + T_{\text{Decoder}} = 647350 + 109490 = 756840 \tag{4.5}$$

The overall time taken turns out to be 7,56,840 seconds.

4.5.2 For 50 nodes

Test accuracy is calculated by varying the number of stragglers in the test dataset after training the setup for different cases of stragglers.

A.Test Accuracy

Here during training phase the distributed setup will be trained starting from the case of 0 stragglers upto to the case of 10 stragglers where 10 input partitions are chosen at random & are replaced with -1's in all the chosen parts.

Table 4.16: Trained for 0-10 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	98.09
10	97.35
20	95.13
30	89.38
40	64.95
49	13.19

Table 4.17: Trained for 0-20 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	98.04
10	97.50
20	96.42
30	89.95
40	79.75
49	22.04

Table 4.18: Trained for 0-30 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	98.10
10	97.69
20	96.42
30	94.01
40	84
49	25.20

Table 4.19: Trained for 0-40 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	97.94
10	97.75
20	96.78
30	94.35
40	85.69
49	28.09

Table 4.20: Trained for 0-49 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	97.75
10	97.71
20	96.64
30	94.73
40	85.87
49	30.37

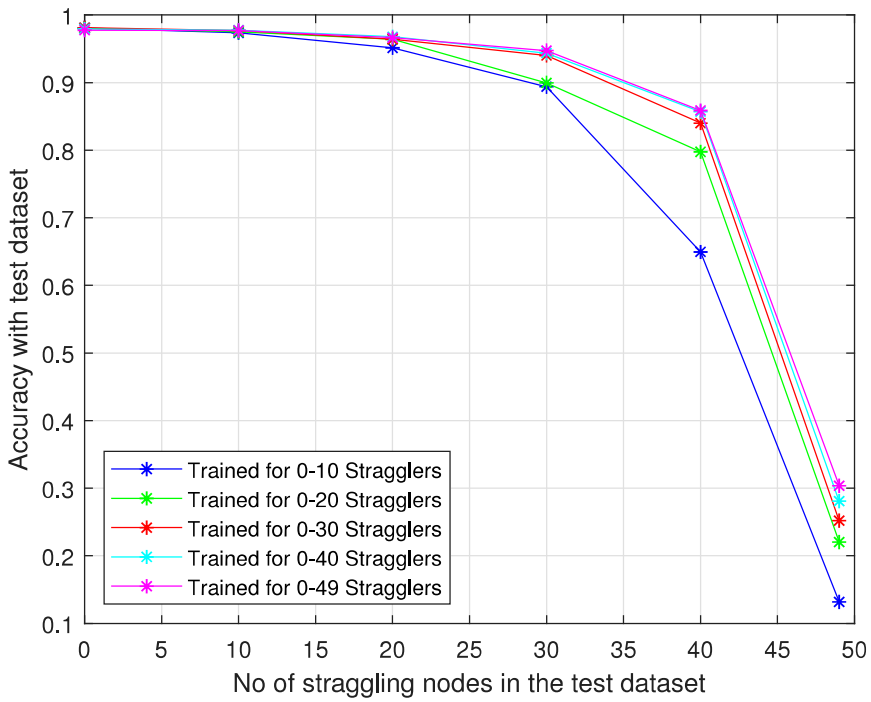

Figure 4.4: Plot of Test Accuracy for 50 nodes

B.Overall Time Taken

- Overall time taken is calculated as a function of input size(row size) & the prediction vector length in each CNN.

- Time taken at each node is proportional to the computational complexity(flops) at each node.All nodes are assumed to have same computational capacity.

- The overall time taken is made equal to the case of 10 nodes by the varying the prediction vector length in each CNN.

- Overall time for this case is calculated & is equated to the value obtained from eq. 4.5.The prediction vector length turns out to be 9 .

- The overall time taken turns out to be 770841 seconds.

4.6 Conclusion

We can observe from Fig.4.3 & Fig.4.4 that when the distributed setup is getting trained for more number of stragglers cases during inference the test accuracy values in general follow an increasing trend making the whole system robust to stragglers.

CNN architecture yields better accuracy values for less number of stragglers in the test dataset when compared with its counterpart MLP-architecture. Whereas MLP architecture performs better for more stragglers in the test dataset.

4.7 Comparison

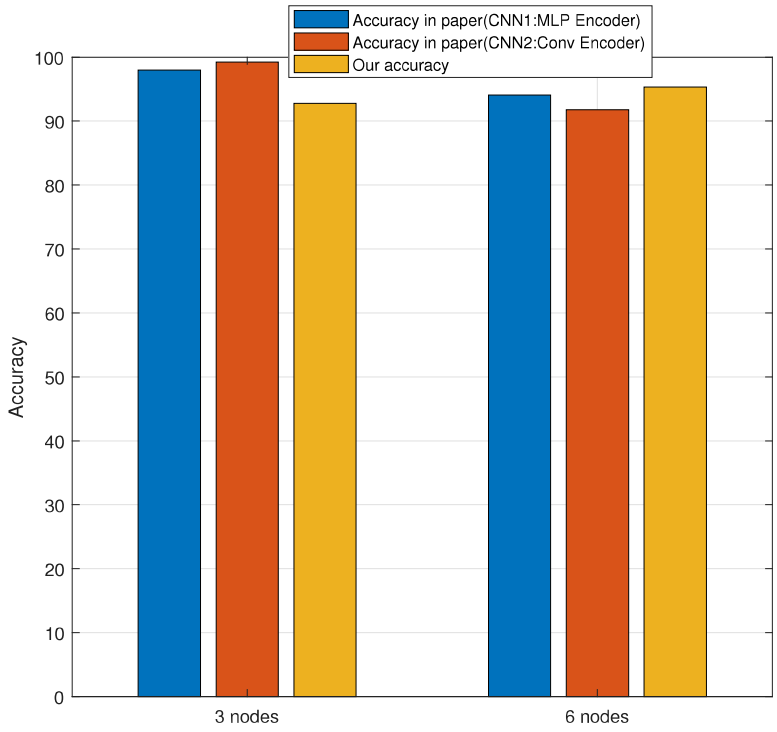

Figure 4.5: Comparison with [2]

- Comparison is with respect to MNIST dataset.
- Two cases are considered one for 3 & another one for 6 nodes.Each node follows CNN architecture.
- Loss function considered is Cross Entropy.
- As the number of nodes increase,our distributed setup gives better accuracy compared to its counterpart in [2].
- For both cases the distributed setup is trained for exactly one of the node being a straggler.Also during inference the is distributed setup is tested for exactly one out of all nodes being a straggler.
- The CNN model used in each node is ResNet-18 which consists of 18 layers .But the architecture of CNN considered considered in our case is simple consisting of 5 Layers.Using this simpler architecture we got better accuracy when compared with their counter parts when the number of working nodes is 6.

4.8 Experimentation On Fashion MNIST Dataset

Fashion MNIST is collection of Zolando's article images consisting of 60000 training examples & 100000 test examples.Similar to MNIST these images are greyscale with size 28×28.The images in the dataset belong to 10 classes.

Every image present in the dataset is partitioned .The images obtained after partitioning are given as input to each CNN present in the distributed setup.

4.8.1 Architecture

A .For 10 Nodes

The architecture of each CNN & as well as decoder is for the case of 10 & 50 nodes is presented in this section.

Table 4.21: Architecture Of Each CNN

Layer	Output Size	Parameters	Activation
Input	(None,3,28,1)	0	-
1^{st}Conv2D	(None,3,28,32)	832	Relu
Maxpooling 2D	(None,2,14,32)	0	-
2^{nd}Conv2D	(None,2,14,64)	51264	Relu
Maxpooling 2D	(None,1,7,64)	0	-
Flatten	(None,448)	0	-
FullyConnected	(None,100)	44900	Relu

Table 4.22: Architecture Of Decoder

Layer	Output Size	Parameters	Activation
Input	(None,1000)	0	-
FullyConnected	(None,500)	500500	Relu
FullyConnected	(None,10)	510	Softmax

B .For 50 Nodes

Table 4.23: Architecture Of Each CNN

Layer	Output Size	Parameters	Activation
Input	(None,1,28,1)	0	-
1^{st}Conv	(None,1,28,32)	832	Relu
Maxpooling 2D	(None,1,14,32)	0	-
2^{nd}Conv	(None,1,14,64)	51264	Relu
Maxpooling 2D	(None,1,7,64)	0	-
Flatten	(None,448)	0	-
FullyConnected	(None,52)	22899	Relu

Table 4.24: Architecture Of Decoder

Layer	Output Size	Parameters	Activation
Input	(None,2600)	0	-
FullyConnected	(None,500)	1300500	Relu
FullyConnected	(None,10)	510	Softmax

4.8.2 Training Parameters

- Batch Size : 200
- Learning Rate : 0.0001
- Training DataSet Size : 1,20,000
- Optimizer : Adam
- Loss Function : Cross Entropy
- Number of epochs for training as well as testing : 300

The size of kernels & biases for different layers and their distributions are shown in Table 4.5

Table 4.25: Weights and their distributions for convolutional layers

Item	Layer	Size	Distribution
Weights	1^{st}Conv	(5,5,1,32)	Truncated Normal with mean = 0 & standard deviation = 0.07
Weights	2^{nd}Conv	(5,5,32,64)	Truncated Normal with mean = 0 & standard deviation = 0.07
Biases	1^{st}Conv	(1,32)	Truncated Normal with mean = 0 & standard deviation = 0.05
Biases	2^{nd}Conv	(1,64)	Truncated Normal with mean = 0 & standard deviation = 0.05

- Truncated Normal Distribution is drawn from [-2,2].
- Weights for fully connected layers in each CNN as well as decoder are initialized are initialized according to truncated normal distribution with mean 0 & standard deviation 0.003.
- Biases for fully connected layers in each CNN as well as decoder are initialized are initialized according to truncated normal distribution with mean 0 & standard deviation 0.006.

4.8.3 Training Process

The distributed setup is trained for different possible cases of stragglers. If the setup is to be trained for h straggles , then after appending the required number of zeros(zero padding) & partitioning h input partitions are chosen at random with uniform distribution out of total input partitions & all pixel values in these chosen parts are replaced with -1's. The size of the training set is increased to cover more number of straggler cases.

4.9 Results & Discussion

Here i present the results of test accuracy versus number of straggling nodes in the test set for the distributed setup trained for different cases of stragglers.

4.9.1 For 10 Nodes

Test accuracy is calculated by varying the number of stragglers in the test set.

Table 4.26: Trained for 0-1 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	89.20
1	87.13
2	85.32
3	81.33
4	74.44
5	65.88
6	56.62
7	41.02
8	28.54
9	15.02

Table 4.27: Trained for 0-2 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	88.20
1	87.58
2	86.29
3	83.35
4	81.91
5	78.18
6	71.06
7	65.14
8	53.50
9	34.25

Table 4.28: Trained for 0-3 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	88.35
1	87.13
2	85.90
3	84.52
4	81.72
5	79.08
6	76.56
7	71.11
8	51.96
9	46.97

Table 4.29: Trained for 0-4 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	87.58
1	87.07
2	85.01
3	85.04
4	82.56
5	81.11
6	78.27
7	74
8	66.21
9	50.22

Table 4.30: Trained for 0-5 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	87.51
1	86.61
2	85.32
3	83.52
4	83.06
5	81.19
6	78.75
7	75.95
8	68.62
9	56.49

Table 4.31: Trained for 0-6 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	87.05
1	85.39
2	85.14
3	84.74
4	82.58
5	81.44
6	78.86
7	74.55
8	68.76
9	58.77

Table 4.32: Trained for 0-7 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	87.05
1	85.39
2	85.14
3	84.74
4	82.58
5	81.44
6	78.86
7	74.55
8	68.76
9	58.77

Table 4.33: Trained for 0-8 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	85.57
1	83.25
2	81.29
3	80.73
4	80.27
5	77.66
6	77.36
7	76.13
8	69.85
9	60.80

A. Plot Of Test Accuracy

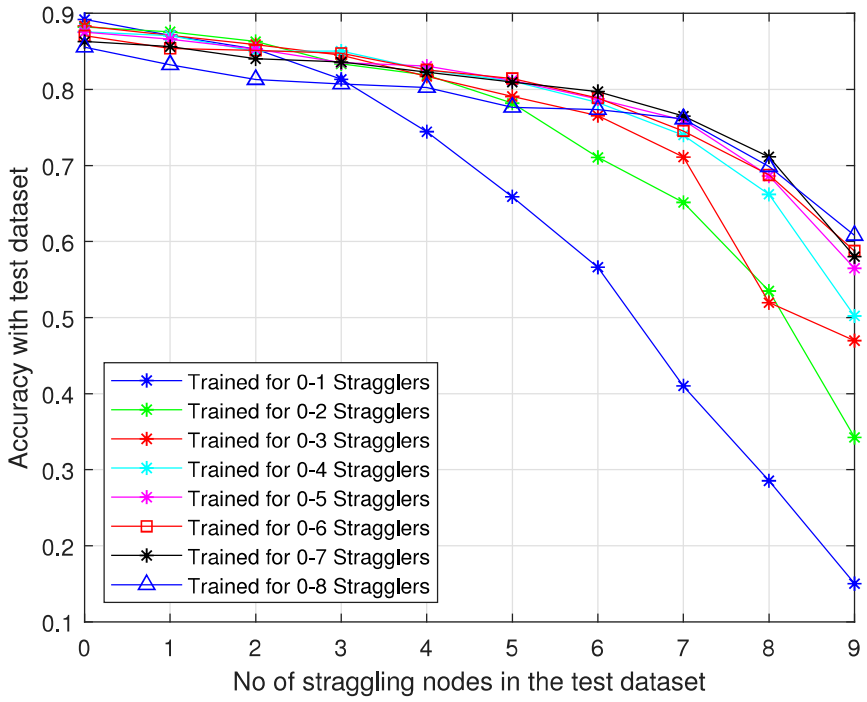

Figure 4.6: Plot of Test Accuracy for 10 Nodes

B. Overall Computation Time

The overall computation time is calculated according to 4.1.

- Time taken by each node T_{node} is calculated as follows:
- Without loss of generality we assume $\alpha = 1$.(Time taken per flop is 1 sec)

$$T_{\text{1stConv}} = 3 \times 28 \times 32 \times [2(5^2) - 1] = 131712$$
$$T_{\text{1stmaxpool}} = 4 \times 2 \times 14 \times 32 = 3584$$
$$T_{\text{2ndConv}} = 2 \times 14 \times 64 \times [2(5^2)(32) - 1] = 2865408$$
$$T_{\text{2ndmaxpool}} = 2 \times 14 \times 64 = 1792$$
$$T_{\text{fullyconnected}} = 100 \times [448 + 448 - 1] = 89500$$

$$T_{\text{Node}} = T_{\text{1stConv}} + T_{\text{1stmaxpool}} + T_{\text{2ndConv}} + T_{\text{2ndmaxpool}} + T_{\text{fullyconnected}} = 3091996 \quad (4.6)$$

- Time taken by master neural network $T_{\text{Master Neural Network}}$ is calculated as follows :

$$T_{\text{1stFullyConneceted}} = 500 \times [2(1000) - 1] = 999500$$
$$T_{\text{2ndFullyConnected}} = 10 \times [2(500) - 1] = 9990$$

$$T_{\text{Master Neural Network}} = T_{\text{1stFullyConnected}} + T_{\text{2ndFullyConnected}} = 1009490$$

$$T_{\text{Overall}} = T_{\text{Node}} + T_{\text{Master Neural Network}} = 3091996 + 1009490 = 4101486$$

4.9.2 For 50 Nodes

Here also the test accuracy is plotted for various cases of stragglers present in the test dataset after training the setup for stragglers.

Table 4.34: Trained for 0-10 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	88.10
10	86.62
20	85.39
30	78.13
40	60.53
49	16.85

Table 4.35: Trained for 0-20 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	87.62
10	86.18
20	85.10
30	82.77
40	73.00
49	22.75

Table 4.36: Trained for 0-30 Stragglers

Stragglers in Test Set	Test Accuracy(%)
0	86.26
10	84.50
20	83.63
30	81.70
40	76.33
49	40.45

A. Plot Of Test Accuracy

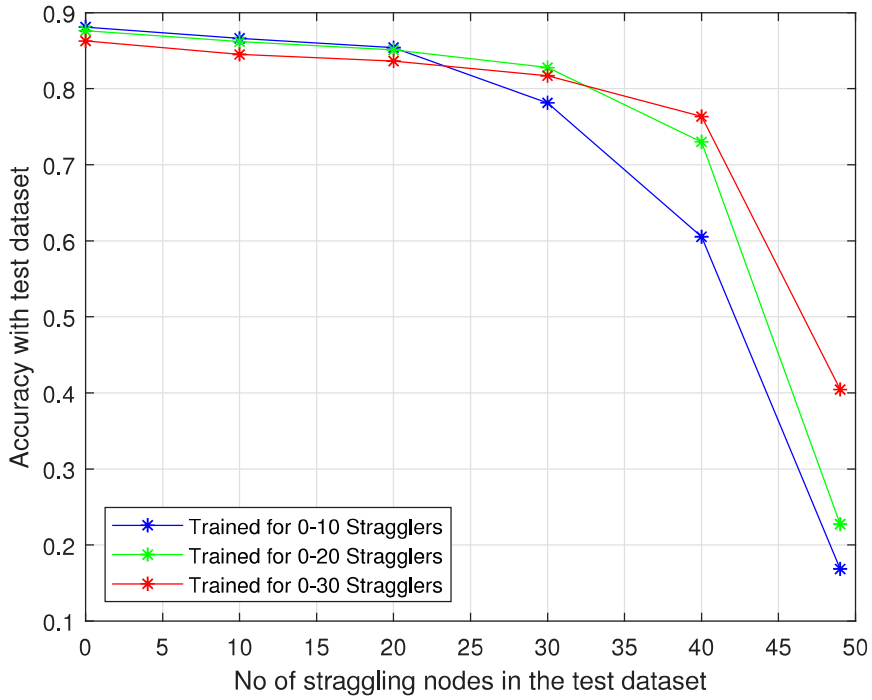

Figure 4.7: Plot of Test Accuracy for 50 Nodes

B. Overall Computation Time

The overall computation time is calculated according to 4.1.

- Time taken by each node T_{node} is calculated as follows:

$$T_{\text{1stConv}} = 1 \times 28 \times 32 \times [2(5^2) - 1] = 43904$$
$$T_{\text{1stmaxpool}} = 4 \times 1 \times 14 \times 32 = 1792$$
$$T_{\text{2ndConv}} = 1 \times 14 \times 64 \times [2(5^2)(32) - 1] = 1432704$$
$$T_{\text{2ndmaxpool}} = 4 \times 7 \times 64 = 1792$$
$$T_{\text{fullyconnected}} = 52 \times [448 + 448 - 1] = 46540$$

$$T_{\text{Node}} = T_{\text{1stConv}} + T_{\text{1stmaxpool}} + T_{\text{2ndConv}} + T_{\text{2ndmaxpool}} + T_{\text{fullyconnected}} = 1526732 \quad (4.7)$$

- Time taken by master neural network $T_{\text{Master Neural Network}}$ is calculated as follows :

$$T_{\text{1stFullyConneceted}} = 500 \times [2(2600) - 1] = 2599500$$
$$T_{\text{2ndFullyConnected}} = 10 \times [2(500) - 1] = 9990$$

$$T_{\text{Master Neural Network}} = T_{\text{1stFullyConnected}} + T_{\text{2ndFullyConnected}} = 2609490$$

$$T_{\text{Overall}} = T_{\text{Node}} + T_{\text{Master Neural Network}} = 1526732 + 2609490 = 4136222$$

4.9.3 Conclusion

From Fig.4.6 we can observe that the distributed setup can tolerate upto 4 stragglers in the test dataset i.e, the test accuracy values are above 80%. As the setup gets trained for more stragglers it becomes robust to stragglers.From Fig.4.7 we can observe that setup classifies the images with good accuracy up to 30 stragglers introduced in the test dataset.We can say that the setup is robust to 30 stragglers.

Chapter 5

Future Scope

The datasets considered were MNIST & Fashion-MNIST which are grey-scale images with not many features.So the architectures considered for both MLP & CNN are very simple.Experimentation can be done by considering complex datasets like CIFAR-10,CIFAR-100 & ImageNet. These datasets consists of color images with RGB channels containing more number of features.

Challenges

- For complex datasets merely increasing layers doesn't increase the accuracy,residual connections are to be used between the layers.
- Increasing the layers in each node increases the time taken for training the entire distributed setup.
- For a given communication bandwidth ,decoder complexity & prediction vector length a good architecture should be selected.

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[2] Jack Kosaian, KV Rashmi, and Shivaram Venkataraman. Learning a code: Machine learning for approximate non-linear coded computation. *arXiv preprint arXiv:1806.01259*, 2018.

[3] Kangwook Lee, Maximilian Lam, Ramtin Pedarsani, Dimitris Papailiopoulos, and Kannan Ramchandran. Speeding up distributed machine learning using codes. *IEEE Transactions on Information Theory*, 64(3):1514–1529, 2017.

[4] Shu Lin and Daniel J Costello. *Error control coding*. Pearson Education India, 2001.

[5] Krishna Giri Narra, Zhifeng Lin, Ganesh Ananthanarayanan, Salman Avestimehr, and Murali Annavaram. Collage inference: Tolerating stragglers in distributed neural network inference using coding. *arXiv preprint arXiv:1904.12222*, 2019.